VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozpoznávání lidských činností pomocí detekce anomálií

Human Action Recognition using Anomaly Detection

2020 Ondřej Stehlík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Stehlík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Rozpoznávání lidských činností pomocí detekce anomálii
Human Action Recognition using Anomaly Detection

Jazyk vypracování: čeština

Zásady pro vypracování:

V poslední době je věnována značná pozornost výzkumu v oblasti rozpoznávání činností. Úspěšné metody mohou být využity v mnoha aplikacích, jako je bezpečnost, analýza lidského chování apod. Je proto potřeba vyvíjet spolehlivé metody pracující v reálném čase. V určitých případech není možné využít některé typy dat pro trénování a následného rozpoznávání neobvyklého či nestandardního chování osob. Pro tyto případy je potřeba se zaměřit na vytvoření modelu standardního chování osob. Jakmile se nějaká osoba vymyká standardnímu chování je vhodné toto označit za anomálii.

Požadavky pro splnění:
1. Seznamte se se získáním pózy kostry z RGB obrazů nebo/a hloubkových map (využít lze OpenPose a také předem získané pózy koster u datasetů).
2. Seznamte se s metodami detekce a klasifikace akcí založené na kosterních příznacích ve video sekvencích.
3. Seznamte se s detekcí anomálií a minimálně jednu metodu využijte.
4. Tyto metody náležitě popište.
5. Experimentálně ověřte funkčnost algoritmů. Případně navrhněte a implementujte vylepšení.
6. Zjištěné poznatky řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

[1] JUNEJO, I N, E DEXTER, I LAPTEV a Patrick PÉREZ. View-Independent Action Recognition from Temporal Self-Similarities. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2011, 33(1), 172-185. DOI: 10.1109/TPAMI.2010.68. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/5432213/
[2] Ronald Poppe: A survey on vision-based human action recognition, Image and Vision Computing, Volume 28, Issue 6, June 2010, Pages 976-990, ISSN 0262-8856
[3] CAO, Zhe, Tomas SIMON, Shih-En WEI a Yaser SHEIKH. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017, 2017, , 1302-1310. DOI: 10.1109/CVPR.2017.143. ISBN 978-1-5386-0457-1. Dostupné také z: http://ieeexplore.ieee.org/document/8099626/
[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose
[5] CHANDOLA, Varun, Arindam BANERJEE, Vipin KUMAR a Yaser SHEIKH. Anomaly detection. ACM Computing Surveys. IEEE, 2009, 2017, 41(3), 1-58. DOI: 10.1145/1541880.1541882. ISBN 978-1-5386-0457-1. ISSN 03600300. Dostupné také z: http://portal.acm.org/citation.cfm?doid=1541880.1541882

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radek Simkanič, DiS**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 14. května 2020 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 14. května 2020 .....

Rád bych na tomto místě poděkoval mému vedoucímu práce Ing. Radkovi Simkaničovi DiS za pomoc se seznámením s problematikou tématu a za konzultace při řešení práce. Dále bych chtěl poděkovat všem, kteří mi jakkoliv pomohli během studia.

Abstrakt

Metody pro detekci anomálií jsou dobře popsány jak u statických tak i dynamických dat různých formátů, avšak u lidských činností tomu tak není. Každý člověk vykonává činnosti odlišně, a to je problém detekce anomálie, když stejná činnost je různými lidmi vykonávána jinak. Například u mávání může ruku zvednou méně či více, nebo může mávat v rychleji či pomaleji.

Tato práce se zabývá detekci anomálií u 4 datasetů reálných aktivit za použití neuronové sítě. U použitých datasetů se provedl základní statistický rozbor, kde se zjišťovaly délky aktivit podle počtů snímků, jejich průměry, mediány a počet aktivit rozdělených podle množství snímků.

Klíčová slova: Anomálie, neuronová síť, strojové učení, detekce anomálií

Abstract

Methods for detecting anomalies are well described for both static and dynamic data of various formats, but this is not the case for human activities. Each person performs different activities and this is the problem of anomaly detection when the same activity is performed differently by different people. For example, in waving, a hand may be raised less or more, or it may wave at a faster or slower rate.

This work deals with anomaly detection in 4 datasets of real activities using neural network. For the used datasets a basic statistical analysis was carried out to determine the predominant lengths of activities according to the number of frames, their averages, medians and the amount of activities divided to groups according to the number of frames.

Keywords: Anomaly, neural network, machine learning, anomaly detection

Obsah

Seznam použitých zkratek a symbolů

NS	–	Neuronová síť
EKG	–	Elektrokardiogram
MRI	–	Magnetická rezonance
API	–	Rozhraní pro programování aplikací
CPU	–	Centrální procesorová jednotka
GPU	–	Grafická procesorová jednotka
TPU	–	Tenzorová procesorová jednotka
SVM	–	Metoda podpůrných vektorů
OS	–	Operační systém
SDK	–	Sada vývojových nástrojů pro vytváření softwaru
RGB	–	Červená, zelená, modrá - typ technologie používající tyhle 3 barvy u obrazu
MSE	–	Střední kvadratická chyba

Seznam obrázků

Seznam tabulek

1 Úvod

Detekce anomálií je problematika spousty odvětví a hraje roli v mnoha kritických systémech. Může být součástí predikce nebezpečného chování či potenciálně teroristického útoku na měkkých cílech. Takové útoky se dějí velmi rychle a momentálně jim nejde moc zabránit. Dobrý kamerový systém se systémem detekcí anomálií by mohl urychlit reakci potřebných lidí a zmírnit následky těchto činů. Dále detekce anomálií může pomoci s nalezením nemocného člověka ve firmách a zabránění dalšího rozšíření nemocí.

V dopravě může analýza obrazu a detekce anomálií zachytit stav řidiče, který je neslučitelný s řízením, jako je například mikrospánek, infarkt či mrtvička. Další využití je u autonomních aut, které analyzují cestu před sebou a vyhýbají se překážkám.

Ve zdravotnictví je detekce anomálií velmi důležitá, jelikož může objevit abnormální chování orgánů, například EKG (Elektrokardiogram) monitoruje stav srdce pacienta. Problém se srdcem se díky tomu dá rychle najít a řešit. MRI (Magnetická rezonance) dokáže najít i anomálie u struktury kostí, orgánů a měkkých tkání.

Tato práce je rozdělena do několika částí. V první části jsou popsány informace ohledně použitého programovacího jazyka Python, jeho distribuce Anaconda, neuronové sítě Tensorflow, zařízení Kinect a použitých datasetů (UTKinect-Actions3D, Florence 3D Actions, JHMDB a MSR Acrions3D).

Další část se zabývá popisem detekce anomálií a rozpoznání aktivit za použití strojového učení. Je zde popsáno k čemu slouží příznaky a pár běžných metod používaných u rozpoznání aktivit (klasifikaci) a detekce anomálií (SVM, K-nejbližší sousedství, K-průměrů, Autoenkodér, Konvoluční neuronová síť).

Poslední část se zabývá popisem programu a testování použitých metod pro rozpoznání aktivit a detekce anomálií. Cílem je implementace a ověření funkčnosti rozpoznání aktivit a detekce anomálií na použitých datasetech.

2 Technologie a základní informace

V této práci byl použit k implementaci algoritmu programovací jazyk Python. Je to interpretovaný, objektově orientovaný programovací jazyk s vysokou úrovní dynamické sémantiky. Díky vysoké úrovni datových struktur, kombinované s dynamickými typy a dynamickými vazbami, je velmi atraktivní pro rychlý vývoj aplikací, pro použití jako skriptovací jazyk, nebo jako jazyk který propojuje existující komponenty dohromady. Jednoduchá a snadno naučitelná syntaxe Pythonu zdůrazňuje čitelnost, a proto snižuje náklady na údržbu programu. Python podporuje moduly a balíčky, které podporují modularitu programu a opakované použití kódu. Interpret Python a jeho rozsáhlá standardní knihovna je dostupná ve zdrojové nebo binární podobě bez poplatku pro všechny hlavní platformy, kterou jde volně distribuovat [1].

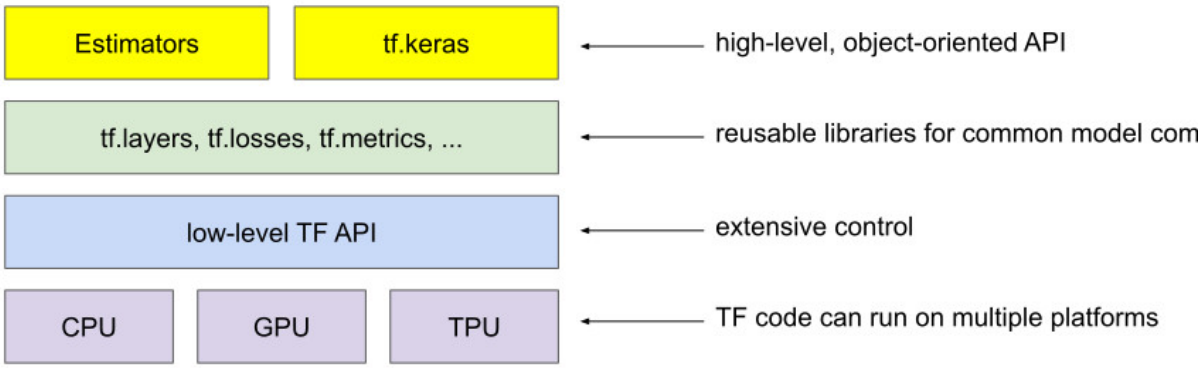
K nasazení jazyka Python byla použita Anaconda [2], která je klíčová pro implementaci neuronové sítě Tensorflow. Anaconda je svobodný a otevřený software [3], fungující jako distribuce programovacích jazyků Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN a další. Využívá se v datové vědě, strojovém učení, zpracování dat velkých rozměrů, prediktivní analytice atd. Tento software slouží pro správu a nasazení různých balíčků. Verze balíčků jsou řízeny správcem balíčků Conda [4]. Anaconda distribuce obsahuje kolem 1500 balíčků, které jsou kompatibilní s operačními systémy Windows, macOS a Linux.

Největší rozdíl mezi Conda a běžným Python správcem balíčků pip [5] je způsob řízení závislostí balíčků, který je významným aspektem pro datovou vědu a důvod proč Conda existuje. Když pip instaluje balíčky, tak automaticky instaluje všechny závislé balíčky, aniž by zjišťoval, zda-li to není ve sporu s již nainstalovanými balíčky. Například uživatel který má pomocí pip nainstalovaný Tensorflow, může mít problém nainstalovat nějaký nový balíček, který vyžaduje odlišnou verzi numpy knihovny, oproti té kterou využívá Tensorflow. Takže nastává problém, kdy se nainstaluje verze balíčku, s kterou Tensorflow nedokáže pracovat. Druhá možnost je, že se balíček nainstaluje, ale produkuje odlišné výsledky. Conda analyzuje celé prostředí, zpracovává jak nainstalovat kompatibilní set závislostí a v případě, kdy nejde nainstalovat kompatibilní set upozorní uživatele chybou [6].

2.1 Neuronová síť Tensorflow

Tensorflow je svobodná a otevřená [3] softwarová knihovna pro diferencovatelné programování a programování datového toku napříč širokou škálou úloh. Je to symbolická knihovna, která je také použita pro aplikace strojového učení, jako jsou neuronové sítě. Tensorflow byl vyvinut týmem od společnosti Google. Verze 1.0.0 byl vydána 11.2.2017 a verze 2.0 v září 2019. Tensorflow má flexibilní architekturu, která umožňuje lehké nasazení na různých platformách, jako je CPU (centrální procesorová jednotka), GPU (grafická procesorová jednotka) a TPU (tenzorová procesorová jednotka) od desktopů přes clustery serverů až po mobilní zařízení [7]. TPU (Tensor processing unit) je speciálně navržený integrovaný obvod pro strojové učení pomocí neuronových sítí, vyvinutý společností Google [8]. Hierarchii Tensorflow nástrojů lze vidět na obrázku 1.

Základem je kód, který běží přímo na hardwaru. Nahoru po hierarchii postupují nástroje, které jsou více uživatelsky přívětivé [9].



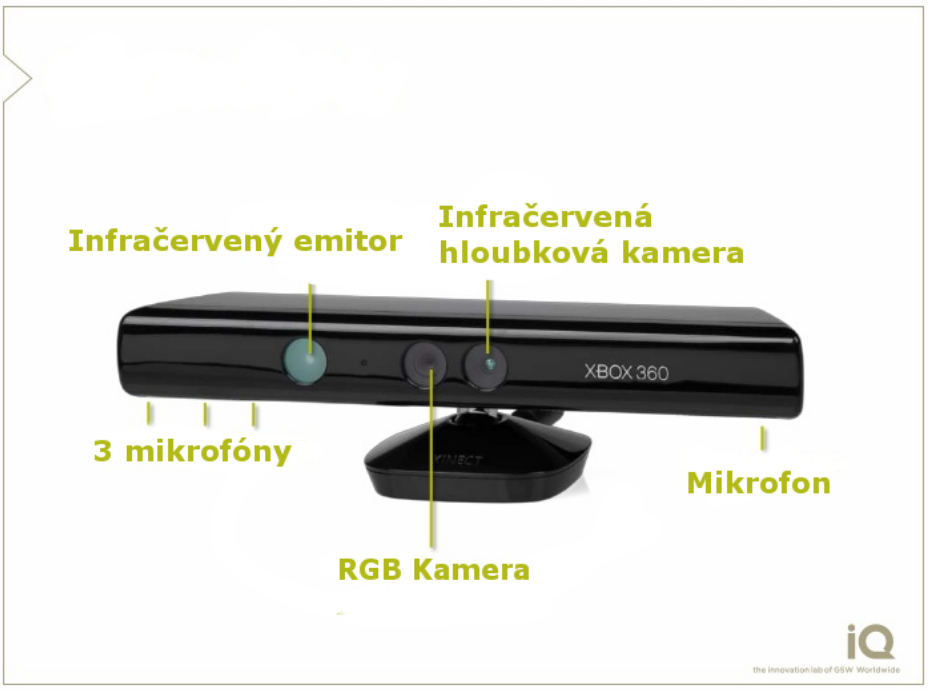
Obrázek 1: Hierarchie Tensorflow nástrojů [10].

Nad sítí Tensorflow bylo použito **API Keras**, které bylo navrženo pro zjednodušení uživatelského ovládání sítě Tensorflow. Keras následuje ověřenou praxi a snižuje kognitivní zatížení uživatele. Nabízí konzistentní a jednoduché API (programovací rozhraní), minimalizuje nutný počet akcí u běžných případů využití a poskytuje srozumitelnou zpětnou vazbu při chybě vytvořené uživatelem. Tohle dělá z Keras jednoduchý a lehce naučitelný nástroj a umožňuje uživateli být více produktivní. S Keras je možno použití více backendů (páteřní část programu) a enginů (sada nástrojů pro tvorbu softwaru). Důležitou vlastností je, že je přenositelný mezi těmito enginy. Uživatelem vytvořený model pro daný backend lze přesunout na odlišný backend a model si stále zachová funkčnost. Dostupné backendy pro Keras jsou Tensorflow, CNTK a Theano. Keras také podporuje použití více GPU najednou [11].

2.2 Kinect

Kinect je zařízení, které snímá lidský pohyb a zvuk. Umožnil lidem ovládat počítač pomocí pohybů a hlasu. Byl vytvořeny společností Microsoft pro herní konzole Xbox360, Xbox One a pro počítače s operačním systémem Windows.

První generace byly vydány pro herní konzoli Xbox360 roku 2010. Verze pro počítače s OS Windows měla počátky už začátkem roku 2011, ale až na jaře roku 2012 byl vydán Kinect SDK (sada nástrojů pro vytváření softwaru) pro Windows. V roce 2013 byla vydána nová vylepšená verze Kinectu pro herní konzoli Xbox One a rok na to bylo možno pořídit Windows verzi nového Kinectu. V roce 2017 Microsoft ukončil výrobu všech předchozích Kinectů, avšak rok na to oznámili, že vyvíjí Azure Kinect, který je mířený pro podnikový software, umělou inteligenci a je vybudován okolo Microsoft Azure, což je jejich cloudová platforma (služba dostupná z internetu). Nový Azure Kinect tedy není součástí herní konzole, ale od roku 2019 jej lze pořídit samostatně [12]. Na obrázku 2 jsou znázorněny jednotlivé části prvního Kinectu.



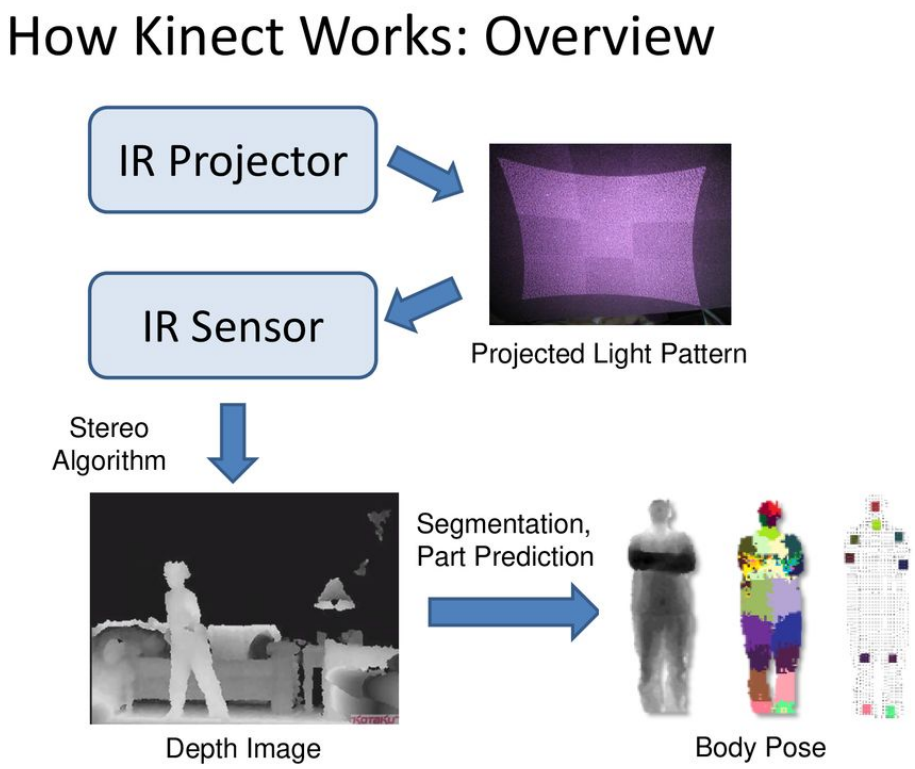
Obrázek 2: Struktura Kinectu [13].

Kamera Kinectu je typu RGB, to znamená, že snímá 3 hlavní barvy, a to je červená, zelená a modrá. U prvního Kinectu má tato kamera pixelové rozlišení 640 × 480 a snímkovací frekvenci 30 snímku/s. Kinect 2 má rozlišení této kamery 1920 × 1080 a stejnou snímkovací frekvenci. Tato kamera napomáhá při rozpoznávání tváří a těl[12].

Hloubková kamera obsahuje monochromatický **CMOS senzor** a **infračervený emitor**, pomocí kterých je vytvářená hloubková mapa nahrávané místnosti. Hloubková mapa udává vzdálenost objektů od kamery. Emitor vyšle oku neviditelné infračervené paprsky. Tyto paprsky se vracejí na CMOS senzor a čím déle cestují prostorem, tím více se jich ztrácí. Ta oblast na senzoru, kde je množství paprsků více je blíže kameře a ta oblast kde je paprsků méně je dál od kamery. Z těchto dat je následně vytvořena hloubková mapa. Použití infračerveného světla z části řeší problém s odlišným nasvícením místností, jelikož infračervené světlo není součástí viditelného spektra světla. Například světlo z LED (světlo vyzařující dioda) a kompaktní zářivky má slabou infračervenou složku, takže neovlivní hloubkovou mapu [12]. Proces zachycení hloubkové mapy lze vidět na obrázku 3.

První Kinect má pixelové rozlišení hloubkové kamery 320 × 240, horizontální zorné pole 57° a vertikální zorné pole 43°. Dokáže zachytit 2 kostry najednou, u kterých rozpozná 20 kloubů pro každou kostru. Hloubkovou mapu zvládne udělat pro objekty od 40cm do ~4,5m vzdálené od kamery. Kinect 2 má rozlišení hloubkové mapy 512 × 424, horizontální zorné pole 70° a vertikální zorné pole 60°. Dokáže zachytit až 6 koster najednou a snímat 26 kloubu u každé kostry. Hloubkovou mapu zvládne udělat pro objekty od 40cm do ~4,5m. Další vlastnost kterou

mají všechny Kinecty, je schopnost izolování lidského hlasu od ruchu v pozadí a tím umožnit uživateli ovládat počítač hlasem. Z tohohle důvodu má Kinect několik mikrofónů [14].



Obrázek 3: Zachycení hloubkové mapy [13].

3 Datasety

Dataset je soubor dat, který v tomhle případě obsahuje informace o lidských aktivitách. Datasety použité v téhle práci (UTKinect-Action3D, Florence 3D Actions, JHMDB, MSR) mají data někdy ve více formátech, ale všechny formáty vychází z hloubkových map. Tyto hloubkové mapy byly u všech použitých datasetů pořízeny zařízením Kinect (kapitola 2.2) nebo jemu podobným.

U všech datasetů se použila data ve formátu pozic kloubů lidského těla. Mezi tyto klouby spadají i některé body, které nejsou samy o sobě klouby, ale jsou důležité pro detekci kostry člověka. Jsou to například body středu páteře, krku a hlavy. U datasetu JHMDB jsou 2D (osy x, y) pozice kloubů, u datasetů Florence 3D Actions a UTKinect-Action3D jsou 3D pozice kloubů (osy x, y, z) a u datasetu MSR jsou 3D (osy x, y, z + pravděpodobnost), kde je navíc uvedena pravděpodobnost, s jakou se kloub na daném místě nachází. Osy x a y jsou hodnoty poloh těchto kloubů, jako odchylka od nulového bodu v metrech, kde nulový bod je střed 2D obrazu. Osa z je vzdálenost objektu (v tomhle případě kloubu) od kamery. Čtvrtá dimenze, kterou využil pouze dataset MSR je pravděpodobnost, s jakou se kloub na daném místě nachází. V následující části je popis a základní statistická analýza délky jednotlivých aktivit těchto použitých datasetů.

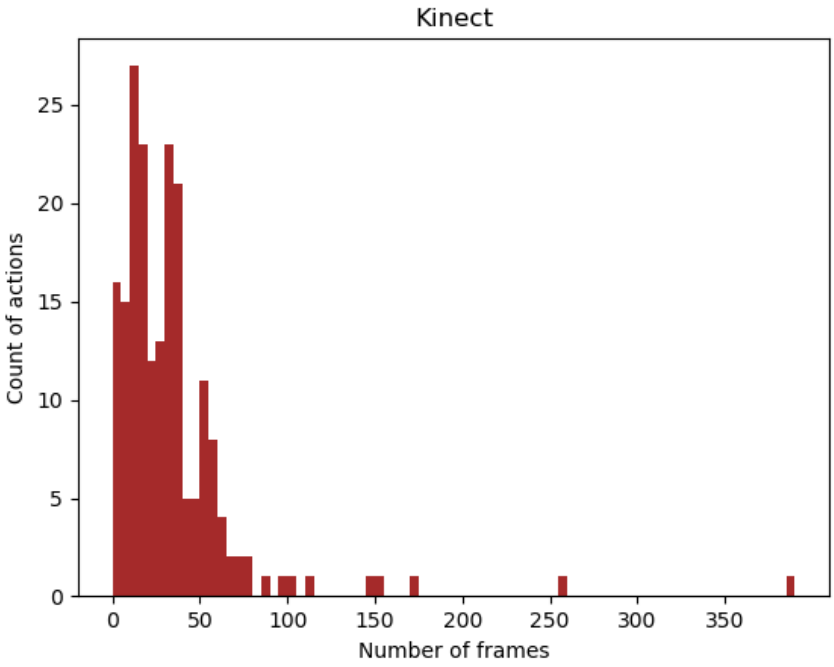
Bylo zjišťováno, jak moc se rozložení délek podobá Gaussově křivce (taky známo jako normální rozdělení). Byla totiž téze, že rozložení těchto délek bude podobné Gaussově křivce, protože patří mezi nejčastější rozložení četností výskytu určitého jevu. Tato téze se do jisté míry ukázala, jako pravda u 3 ze 4 datasetů. Dále byly zkoumány hodnoty mediánu, modu a průměru délek aktivit.

3.1 UTKinect-Action3D dataset

Videa byla u tohoto datasetu zachycena pomocí jednoho stacionárního Kinectu (kapitola 2.2). Je zde obsaženo 10 typů akcí: chůze, posazení se, postavení se, zvednutí se, zvednutí věci, házení, tlačení, táhnutí, mávání rukama a tleskání rukama. Těchto 10 činností vykonávalo 10 subjektů (lidí) a každý vykonal danou činnost dvakrát. Dataset obsahuje celkem 198 akcí (pro 2 akce nejsou data). Byly zaznamenány tři formy dat: RGB, hloubková mapa a pozice kloubů kostry. Všechny tři data jdou zpracovat zvlášť.

Kostry u tohohle datasetu obsahují 20 kloubů a hloubkové mapy byly pořízeny při snímkové frekvenci 30 snímků/s, ale byly použity pouze snímky, kdy byla lidská kostra na snímku úspěšně rozpoznána. Kvůli tomu je finální snímkovací frekvence datasetu kolem 15 snímků/s [15, 16].

Křivka délek aktivit připomíná Gaussovou křivku posunutou doleva. Většina aktivit má délku do 75 snímků, ale je tu pár aktivit, které se délkou vyjímají, ale těchto delších aktivit je jen pár. Na ose X je počet snímků, takže maximální počet snímků u aktivit je 390 a minimální je pod 5 snímků. Graf je rozdělen do částí po 5 snímcích, tak se několik aktivit jeví, jako s délkou 0, ale ve skutečností mají délku pod 5, ale nad 0 snímků. Nejčastější délka, tedy modus je 10 snímků, medián je 25 snímků a průměr je 31,6 snímků.



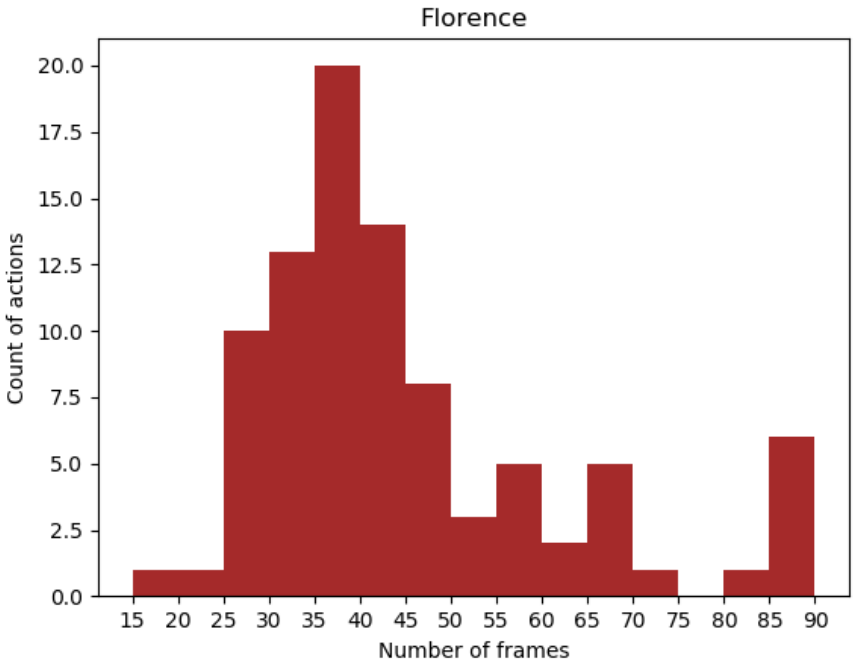
Obrázek 4: Rozdělení aktivit podle počtu snímků Kinect.

3.2 Florence 3D Actions dataset

Dataset byl vytvořen na Florentské univerzitě v roce 2012 a byl pořízen pomocí kamery Kinect (kapitola 2.2). Obsahuje 9 aktivit: mávnutí rukou, pití z láhve, zvednutí telefonu, tleskání, zavázání si tkaniček, sednutí si, zvednutí se, podívání se na hodinky a úklon těla. Tyto činnost vykonalo 10 subjektů (lidí) a každou činnost vykonal subjekt 2 až 3 krát. Výsledkem bylo dohromady 215 vzorků aktivit [17, 18].

Dataset se skládá z krátkých videí ve formátu multimediálního kontejneru AVI a souboru obsahující pozice 15 kloubů kostry těla, kam patří i střed těla, krk a hlava. Snímkovací frekvence tohohle datasetu je 30 snímků/s, stejně jako je u Kinectu.

Rozložení délky aktivit ve snímcích, které je možno vidět na obrázku 5 připomíná Gaussovou křivku s pár extrémními případy větší délky. Také lze vidět, že se většina délek pohybuje kolem stejné hodnoty. Na ose X je počet snímků, takže maximální počet snímků u aktivit je 90 a minimální je 15. Nejvíce aktivit se pohybuje kolem 40 snímků. Průměr snímků pro aktivitu je 42,665, medián je 37,5 a modus je 35 snímků.

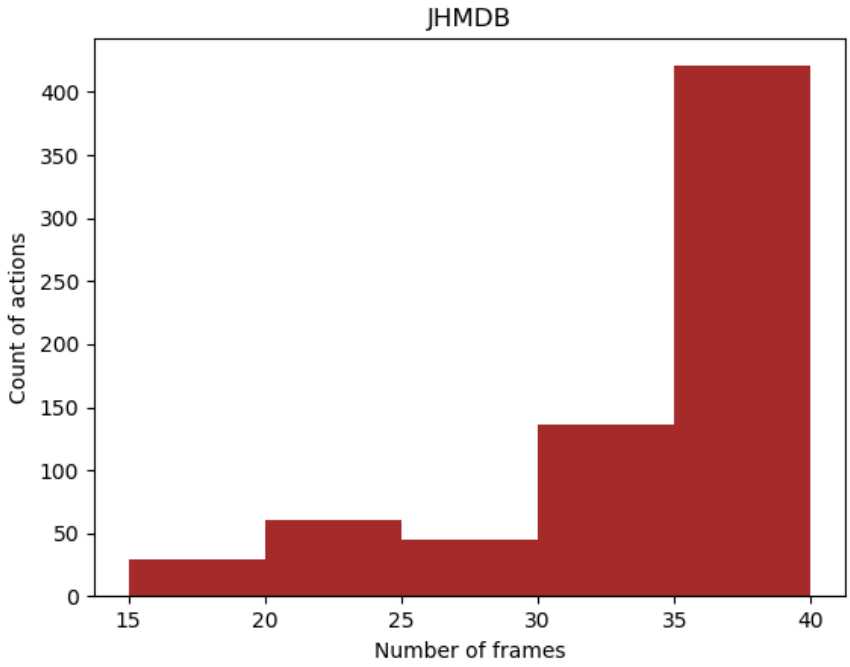


Obrázek 5: Rozdělení aktivit podle počtu snímků pro Florence.

3.3 JHMDB dataset

Tento dataset byl vytvořen anotací většího datasetu HMDB51, týmem lidí pocházejících z několika univerzit v Německu, USA a Francii. Obsahuje více než 5100 klipů 51 kategorií lidských aktivit. Anotovány nebyly všechny aktivity, takže vznikl dataset, který má 21 kategorií aktivit a ty jsou: česání vlasů, chytnutí míčku, tleskání, vyjití schodů, golfový úder, skok, kopnutí míče, zvednutí předmětu, nalití sklenice, shyb, tlačení, běh, hod balónem, střelba z luku, střelba ze zbraně, sednutí si, hod, chůze, mávnutí a máchnutí basebalové pálky. K anotaci byl použit puppet (3D model člověka). Výsledkem je 36-55 klipů pro jednu kategorii s 15-40 snímky. Dataset obsahuje data v těchto formátech: videa, 2D pozice kloubů, puppet mask (maska modelu)a puppet flow (tok modelu). Místo textových souborů pro pozice kloubů byly použity mat soubory používané v MATLABu (matematický programovací jazyk). Tato práce využila formát 2D pozic kloubů, kde počet kloubů pro kostru je 15 [19, 20].

Oproti předchozím datasetům lze vidět, že se rozložení délek vůbec nepodobá Gaussově křivce. Většina klipů je podobné délky a čím kratší jsou, tak tím menší je jejich zastoupení. Naprostá většina délek má kolem 40 snímků. Tomu také odpovídá, že modus i medián má hodnotu 40. Jen průměr se liší a má hodnotu 33,8. Počet všech klipů dohromady je 722, z toho přes 400 je délky 40 snímků.



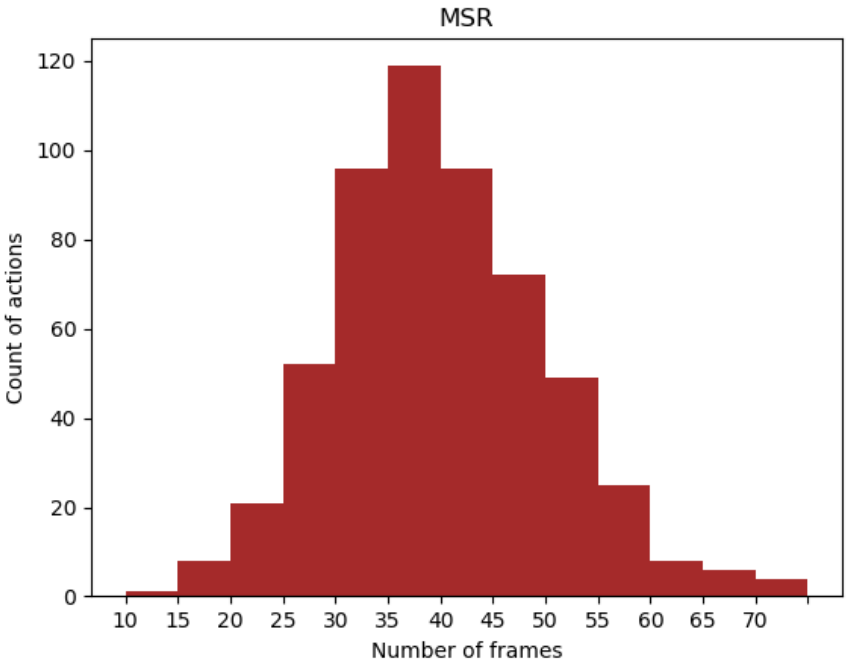
Obrázek 6: Rozdělení aktivit podle počtu snímků JHMDB.

3.4 MSR Action3D dataset

Dataset MSR-Action3D se skládá z hloubkových map lidských činností. Obsahuje dvacet kategorií aktivit: vertikální mávnutí paží, horizontální mávnutí paží, chytání rukou, dopředný úder, vysoký hod, nakreslení x, nakreslení fajfky, nakreslení kružnice, tleskání rukou, mávnutí oběma rukama, boxování ze strany, ohnutí, přední kop, boční kop, běh, tenisová úder, tenisové podání, golfový úder, zvednutí a házení. Těchto 20 aktivit provádělo 10 subjektů (lidí), kteří každou aktivitu provedli 2 až 3 krát. Vytvořil jej Wanqing Li během svého působení v Microsoft Research Redmond (výzkumná laboratoř Microsoftu ve městě Redmond). Hloubkové mapy byly zachyceny ve snímkovací frekvenci 15 snímků/s.

Z těchto hloubkových map udělal Yi Wen Wan z Texaské severní univerzity 4D koordináty kloubů kostry. Bylo zachyceno 20 kloubů/bodů, kde hodnota pro čtvrtou dimenzi, tedy pravděpodobnost se pohybuje mezi 0 a 1, kde 1 je 100% a 0 je 0% [21].

Ze všech datasetů se rozložení délek klipů tohohle datasetu nejvíce podobá Gaussově křivce. Nejdou zde pozorovat téměř žádné deviace od této křivky. Z toho lze odhadnout, že medián, modus ani průměr se nebude moc lišit. Modus i medián má hodnotu 35 snímků a průměr má hodnotu 37,6 snímků. Celkový počet klipů obsažených v datasetu je 557.

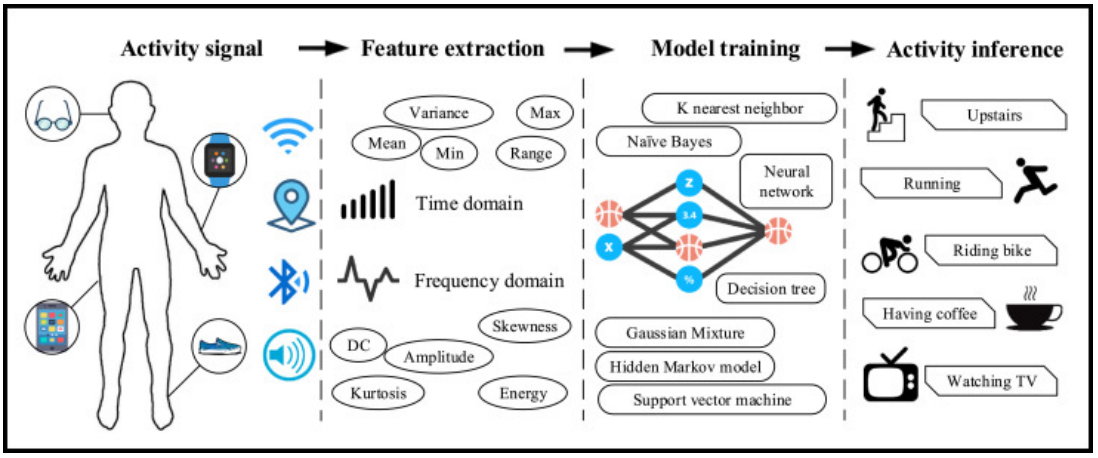


Obrázek 7: Rozdělení aktivit podle počtu snímků MSR.

4 Detekce anomálií a rozpoznávání aktivit

V datové vědě je detekce anomálií detekcí vzácných či netypických objektů a událostí nebo observací, které se výrazně liší od většiny dat. Typické anomálie představují nějaký problém, jako může být bankovní podvod, strukturální porucha, zdravotní potíže nebo chyby v textu. Anomálie bývají označovány jako odlehlé hodnoty, šum, odchylky či výjimky.

Rozpoznávání lidských aktivit dnes používá mnoho zařízení jako jsou chytré hodinky, chytré mobily atd. Dokážou předpovídat jak dlouho osoba dělala jaké aktivity, délku a kvalitu spánku, kolik kalorií osoba spálila atd. Rozpoznání aktivit lze rozdělit do několika kroků, které jsou vyobrazeny na obrázku 8. Nejdřív se získají vstupní data, ze kterých se vytvoří příznaky. Tyto příznaky dále slouží jako vstupní data pro nějaký algoritmus strojového učení, který poskytne výstup, nad kterým se provede klasifikace. Klasifikace a detekce anomálií mají stejný průběh, jen rozdílný poslední krok. U klasifikace se objekt porovnává s již naučenými daty a u detekce anomálií se poměřuje nějaká metrika tohoto objektu s nastavenou hodnotou, která rozděluje normální a anomální data.



Obrázek 8: Kroky u rozpoznávání lidských činností [22].

Detekce anomálií a klasifikace jde ruku v ruce se strojovým učením, které se dělí do 3 hlavních kategorií. Učení s učitelem, učení bez učitele a kombinace učení s učitelem a bez učitele.

Učení bez učitele je metoda strojového učení, která vyhledává dříve nezjištěné vzory v souboru dat, kde tento soubor dat nemá jednotlivé akce označené. Předpokládá se, že většina dat je normální a hledání anomálií se provádí hledáním co nejvíce odlišných dat [23].

Učení s učitelem vyžaduje, aby se data učení skládala ze 2 objektů, a to ze vstupního souboru dat a požadovaného výstupu. Tyto dvojice jsou vektor, který se většinou skládá z příznaků daného objektu a požadované výstupní hodnoty (označení kategorie objektu). Učení s učitelem analyzuje učené data a vytváří odvozenou funkci, která slouží pro mapování nových příkladů. V ideálním případě tento algoritmus zvládne správně určit kategorii pro neoznačený objekt [24].

Kombinace s učitelem a bez učitele používá pro učení malou část označených dat, která je tvořena dvojicemi příznak-označení a velkou část neoznačených příznaků. Použitím malým počtem označených dat lze docílit znatelně lepších výsledků, než při učení bez učitele a částečně se vyhne problému s označováním objektů (nemusí se označovat všechny data) [25].

V následující části jsou popsány příznaky a jedny z nejpoužívanějších metod klasifikace jevů a detekce anomálií. Tyto metody jsou SVM, K-nejbližší sousedství, K-průměrů, Autoenkodér a Konvoluční neuronová síť.

4.1 Příznaky

Ve strojovém učení mají příznaky formu měřitelných vlastností, které se analyzují. Příznak je jakákoliv informace v datech, která se dá měřit a je užitečná pro daný systém. Proto se příznaky mohou velice lišit, podle toho jaká data konkrétní systém zpracovává. Kvalitní zpracování příznaků je klíčové pro strojové učení, protože příznaky představují vstupní data. Například příznak u jednoduchého datasetu může být reprezentován sloupcem tabulky.

U videí které jsou vstupními daty pro klasifikaci lidských činností, se hlavně používají 3 kategorie vstupních dat a těmi jsou RGB/grayscale (barevné nebo černobílé záznamy běžnou kamerou) snímky, hloubkové mapy a zachycení pózy kostry. Účelem příznaků je zredukovat velké množství dat, které je obsaženo ve videu do menšího formátu. Video s aktivitou člověka obsahuje mnoho nepotřebných dat, takže se hodí, když se vyberou pouze ta data, která nesou nějakou užitečnou informaci.

Konkrétně pro získání příznaků u lidských aktivit je problém pozice člověka. Člověk se hýbe a vykonává činnosti v různých pozicích vůči senzorům. Například, kdyby se data pro lidské aktivity nějakým způsobem neupravovala, tak program nerozpozná stejnou aktivitu, která je jednou konána na pravé části snímku a jednou na levé části snímku. Proto je potřeba získat příznaky, které jsou invariantní na pohyb, případně i na natočení. Tohle je jeden z důvodů, proč je u lidských činnosti způsob získání příznak kritický.

4.2 Metody detekce anomálií a rozpoznávání aktivit

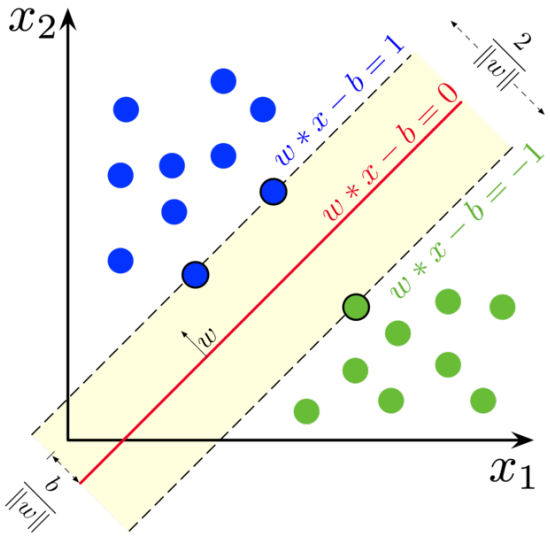
Metody sloužící jako klasifikátory (rozpoznávání) kategorií také často fungují jako detekce anomálií. Proto jsou v této části popsány metody, které fungují jak pro detekci anomálií tak pro klasifikaci kategorií. Pouze při závěrečném vyhodnocování se použije jiná metrika, pomocí které se daný objekt klasifikuje nebo rozpozná, jestli se jedná o anomálii či ne.

4.2.1 SVM - Support Vector Machine

Metoda podpůrných vektorů z anglického Support Vector Machines je metoda strojového učení s učitelem, sloužící zejména pro klasifikaci a také pro regresní analýzu, která se může také použít pro detekci anomálií.

Základem metody SVM je lineární klasifikátor dvou tříd. Cílem úlohy je nalézt nadrovinu, která optimálně rozděluje prostor příznaků tak, že trénovací data náležící odlišným třídám leží v opačných podprostorech. Optimální nadrovina je taková, kde minimální hodnota vzdáleností bodů od roviny je co největší. Jinými slovy, okolo nadroviny je na obě strany co nejširší pruh bez bodů. Česky je tento pruh někdy nazýván jako pásmo necitlivosti nebo hraniční pásmo.

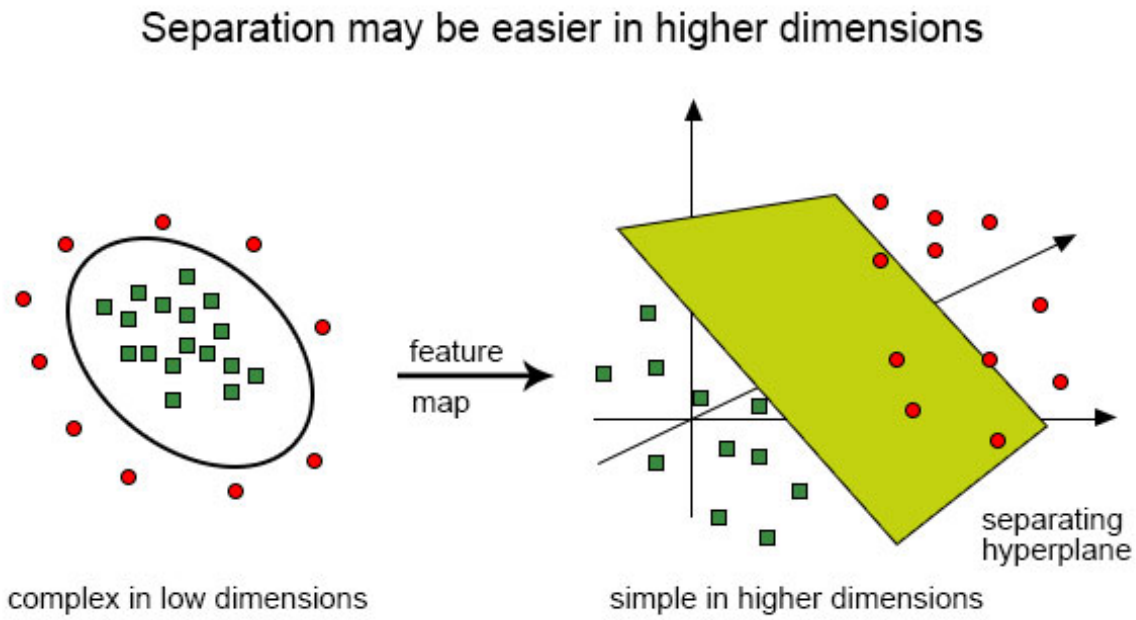
Na popis nadroviny stačí pouze body ležící na okraji tohoto pásma a těch je obvykle málo. Tyto body se nazývají podpůrné vektory, anglicky support vectors. Původní lineární (nadrovina je lineární funkce) algoritmus rozdělující nadroviny byl vytvořen roku 1963. Přišel s ním Vladimir N. Vapnik. Na obrázku 9 je ukázka rozdělující nadroviny a podpůrných vektorů u lineárního SVM [26].



Obrázek 9: Optimální rozdělující nadrovina a hraniční pásmo pro lineární SVM. Body na okrajích pásma jsou podpůrné vektory [27].

Roku 1992 navrhli Bernhard E. Boser, Isabelle M. Guyon a Vladimir N. Vapnik jak vytvořit nelineární SVM klasifikátor za pomocí kernelové metody na rozdělující nadrovině. Je velmi složité vyřešit klasifikaci lineárním klasifikátorem, když je vstup ve tvaru v jakém je na obrázku 10. Červené a zelené body nelze lehce rozdělit lineární funkcí. V tomhle případě se využije kernelova metodu, která transformuje data do vyšší dimenze a je možné rozdělit zelené a červené body jednoduchou rovinou, jakou lze vidět na obrázku 10 napravo. Typů kernelových metod je spousta. Patří mezi ně lineární kernel, polynomiální kernel, Gaussův kernel, exponenciální kernel atd [28].

Je důležité dodat, že pracováním s více dimenzionálním prostorem se zvyšuje chyba generalizace, ovšem když se dodá dostatek vstupních dat, tak SVM stále podává dobré výsledky. Generalizace je měřítko toho, jak je algoritmus schopen klasifikovat pro něj nenaučená data.



Obrázek 10: Přechod příznaků do vyšší dimenze. Nalevo jsou původní příznaky, napravo jsou transformované příznaky [29].

U standardního SVM se rozdělí data jen do 2 tříd, ale je i možnost, jak je rozdělit do více tříd. Dominantní postup, jak tohohle docílit, je rozdělit problém do více binárních klasifikací. Nejčastější přístupy k těmto binárním klasifikacím je metoda, kde se hodnotí třída versus všechny zbylé třídy a druhá metoda je založena na klasifikaci všech tříd mezi sebou [28].

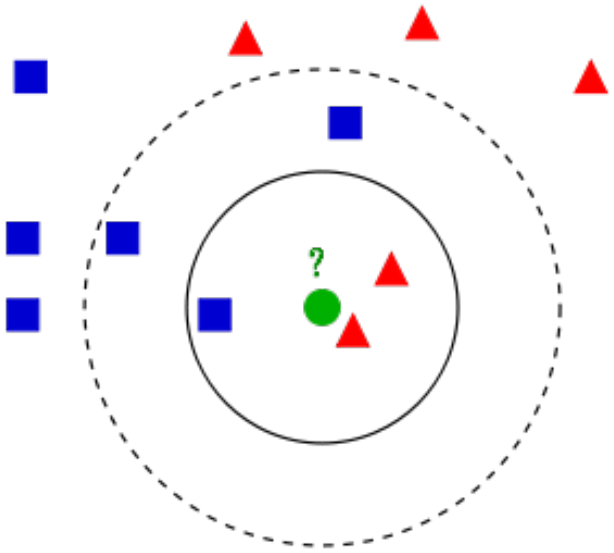
4.2.2 K-nejbližší sousedství

Z anglického pojmu K-nearest neighbors, je metoda patřící mezi metody založené na principu hustoty, kde patří také metody Isolation forest, Local outlier factor atd. Z těchto metod je K-nejbližších sousedství asi nejpoužívanější. Princip hustoty předpokládá, že podobné či stejné jevy se objevují v blízké vzdálenosti od sebe, tedy existují body, kde jsou tyto jevy nahuštěné u sebe. Tato metoda může patřit jak k metodám s učitelem, tak k metodám bez učitele. Pokud se použije tahle metoda ke klasifikaci, tak se hovoří o verzi s učitelem [30, 31].

Princip téhle metody spočívá v nalezení předdefinovaného množství trénovacích objektů/-bodů v nejbližší vzdálenosti od nově vytvořeného bodu a předpovědět označení nového bodu podle označení nejbližších bodů. Proměnná určující množství nejbližších bodů k novému bodu, které se berou v potaz, může být definována uživatelem, nebo se může lišit podle lokálních bodů hustot. Tahle proměnná je označována jako "K"a od toho je odvozeno "K"v názvu této metody. Vzdálenost v jaké má nový bod K nejbližších sousedů může být jakákoliv metrika. Standardně se používá eukleidovská vzdálenost. Tato vzdálenost je použita při detekcí anomálií. Čím je tato vzdálenost větší, tím je daný objekt více anomální. Taky lze nastavit hodnotu vzdálenosti, která

bude rozdělovat objekty na normální a anomální. Metody založené na K-sousedství jsou známy tím, že negeneralizují a prostě se naučí všechna trénovací data [30, 31].

Na obrázku 11 je příklad, jak tato metoda může fungovat. Zelené kolečko je nový bod, u kterého nás zajímá jeho označení (kategorie). Pokud se nastaví proměnná K na 3, tak lze vidět, že 3 nejbližší body jsou 2 červené trojúhelníky a 1 modrý čtverec a kružnice, která reprezentuje vzdálenost bodu od k-nejbližších sousedů je znázorněna plnou čarou. Nový zelený bod bude tedy označen jako červený trojúhelník. Pokud se nastaví parametr K na hodnotu 5, tak nejbližší body jsou 2 červené trojúhelníky a 3 modré čtverce. Zelený bod bude tedy označen jako modrý čtverec a vzdálenost je reprezentována přerušovanou kružnicí na obrázku 11. Pomocí kružnice (vzdálenosti od k-nejbližších sousedů) lze detekovat anomálie. Například když do dané velikosti kružnice nemá bod daný počet sousedů, tak se tento bod označí jako anomálie.



Obrázek 11: Příklad klasifikace K-nejbližší sousedství [32].

I přes svojí jednoduchost je metoda K-nejbližších sousedství úspěšně implementována ve velkém počtu klasifikačních algoritmů, kam patří například klasifikace ručně psaných číslic, nebo klasifikace satelitních snímků [31].

4.2.3 K-průměrů

K-průměrů z anglického K-means je metoda patřící do shlukové analýzy. Ta předpokládá, že shlukované objekty lze chápat jako body v nějakém eukleidovském prostoru, kde počet shluků K je předem dán (případně lze vyzkoušet různá K, pro každé spustit algoritmus znovu a výsledky porovnat). Shluky jsou definovány svými centroidy, což jsou geometrické středy shluků. Objekty se zařazují do toho shluku, jehož centroidu jsou nejblíže. Algoritmus postupuje iterativně tak, že začne s nějakými, obvykle náhodně vybranými centroidy a přiřadí se do nich body. Centroidy se přepočítají tak, aby byly těžištěm shluku bodů, pak se opět přiřadí body k nově stanoveným

centroidům a tak se to dál opakuje, dokud se poloha centroidů neustálí. Detekce anomálií u této metody funguje přes vzdálenosti od centroidů. Když není objekt dostatečně blízko žádnému shluku, tak je považován za anomální. Dále platí, že čím vzdálenější je od nejbližšího shluku, tím je anomálnější [33, 34].

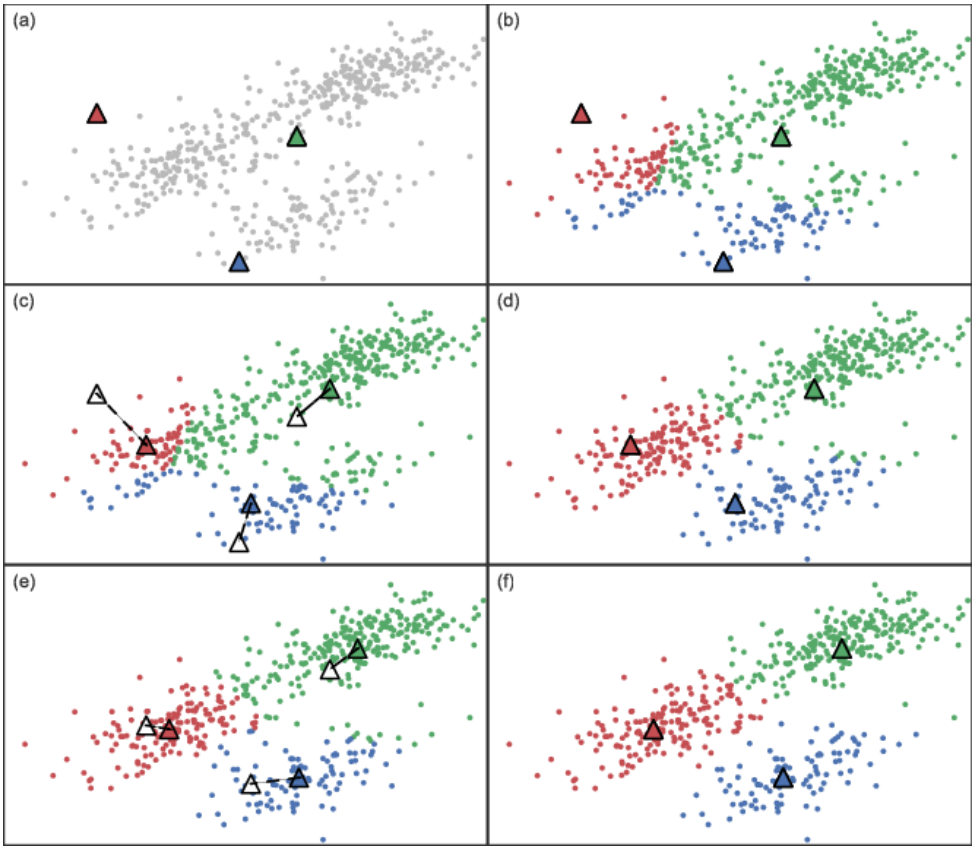
Konvergence algoritmu, tedy stav kdy průměry se rovnají centroidům, je zaručen a v praxi bývá obvykle poměrně rychlý. Nemusí však najít globální optimum ve smyslu nejmenšího součtu čtverců vzdáleností od centroidů a navíc pro různé volby pozic počátečních centroidů mohou vzniknout různá řešení. Klasický algoritmus K-průměrů a jeho varianty konvergují k lokálnímu minimu součtů druhých mocnin vzdáleností mezi bodem a centroidem. Tato konvergence má vzorec, který je popsán na rovnici číslo 1 [33, 34].

$$\sum_{i=1}^{k} \min_{\mu_j \in C} (\|x_i - \mu_j\|)^2 \tag{1}$$

Kde μ_j je průměr, C jsou clustery, x jsou body datasetu a k je počet počátečních bodů průměr.

Tato metoda má spoustu variací, mezi které patří K-medoidů, kde jsou místo bodů průměrů brány body medoidů (medoid je podobný průměru, avšak je omezen na to, aby byl členem datasetu. Používá se tam, kde průměr nebo centroid nemůže být definován.), metoda mediánů, metoda urychlení algoritmu použitím trojúhelníkové nerovnosti atd [34].

Na obrázku 12 je znázorněn průběh algoritmu. Nejdříve vznikne K náhodně zvolených bodů, které fungují jako průměry. V tomhle případě jsou tyto body 3. Dále se vytvoří shluky asociací každého bodu s nejbližším průměrem. Počet shluků je K, stejný jako počet bodů průměrů. V dalším kroku nastává posun bodů průměrů shluků do centroidu daného shluku a přepočet shluků pro každý bod průměrů. Tento krok se opakuje, dokud nenastane konvergence. To znamená, že se průměry rovnají centroidům.



Obrázek 12: Postup algoritmu K-průměrů [35].

4.2.4 Autoenkodér

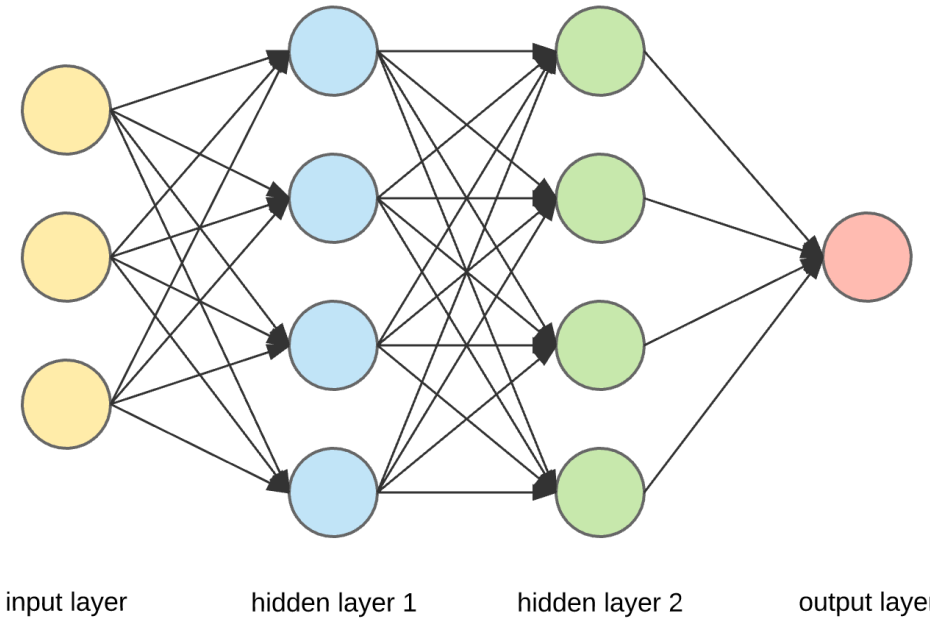
K rozpoznání činností a detekci anomálií se používá spousta druhů neuronových sítí. Jedna z těchto neuronových sítí je označována jako Autoenkodér. Je to metoda učení bez učitele, tedy bez označení akcí. Vstup autoenkondéru má stejný formát, jako má jeho výstup.

Neuronová síť je plně propojená síť, která je tvořena z neuronů a z jejich spojení, což lze vidět na obrázku 13. Shluk neuronů vytváří vrstvy. Každý neuron je propojený se všemi neurony další vrstvy. Neuronová síť je tvořena vstupní vrstvou, skrytými vrstvami a výstupní vrstvou, což lze opět vidět na obrázku 13. Počet skrytých vrstev lze libovolně navyšovat hyperparametrem [36].

Hyperparametr se na rozdíl od ostatních parametrů, které se vytváří během cvičení nastavuje před chodem učícího procesu. Do hypereparametru patří například typologie a velikost neuronové sítě, rychlost učení a velikost batch. Batch je údaj, který označuje kolik dat se vleze do jedné iterace. Iterace se u neuronových sítí většinou označuje jako epocha [37, 36].

Neurony fungují následujícím způsobem. Vezmou vážený součet svých vstupů a ten pošlou nelineární aktivační funkci. Výstup této funkce je výstup neuronu, který se stanem součástí vstupů neuronů další vrstvy. Typů aktivačních funkcí je celá řada a každá vypočítává trochu jiným způsobem.

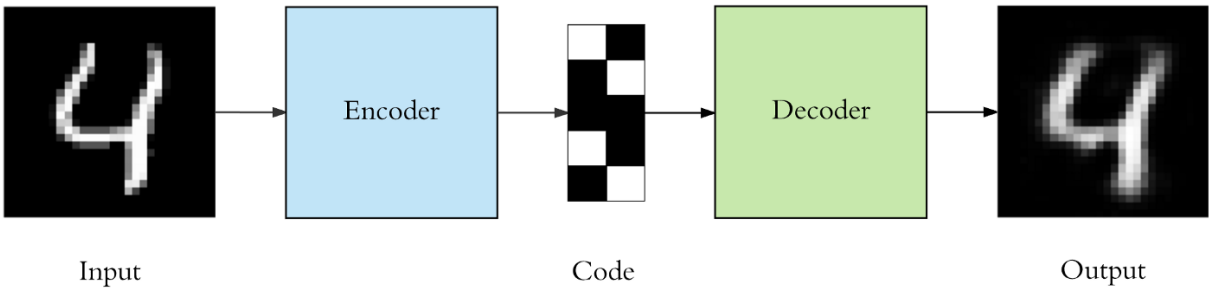
Každý prvek z trénovacích dat musí projít neuronovou sítí. Musí projít všemi neurony a všemi aktivačními funkcemi, až se dostane k výstupu sítě. Poté se poměří tento výstup s původními daty daného prvku za pomocí ztrátové funkce, a tím se získá hodnota chybovosti. Nakonec se provede takzvaná zpětná propagace chyb [36].



Obrázek 13: Popis obecné neuronové sítě [38].

Zpětná propagace chyb je proces, který probíhá zpětně, tedy od výstupní vrstvy po vstupní vrstvu a propaguje chybovost do každého individuálního neuronu. Vypočítá se kontribuce každého neuronu k chybovosti a nastaví se neuronům váhy použitím gradientního sestupu [36].

Autoenkodér se skládá ze 3 hlavních částí: enkodér, kód a dekodér. Enkodér stlačuje data a vyprodukuje kód. Tento kód se snaží dekodér zpátky zrekonstruovat. Tímto vznikne výstup, který je podobný vstupu. Pro vytvoření tohohle algoritmu je potřeba metodu, která vstup enkóduje, metodu která dekóduje a ztrátovou funkci, podle které se výstup porovnává s cílovým objektem. Co se týče označení vstupních dat, tak i když je to metoda bez učitele, tak lze říci, že je sama sobě učitelem. Je to tím, že při učení si tato metoda sama vytváří označení pro naučené objekty. Na obrázku 14 je stručně vyobrazen proces autoenkodéru [39].



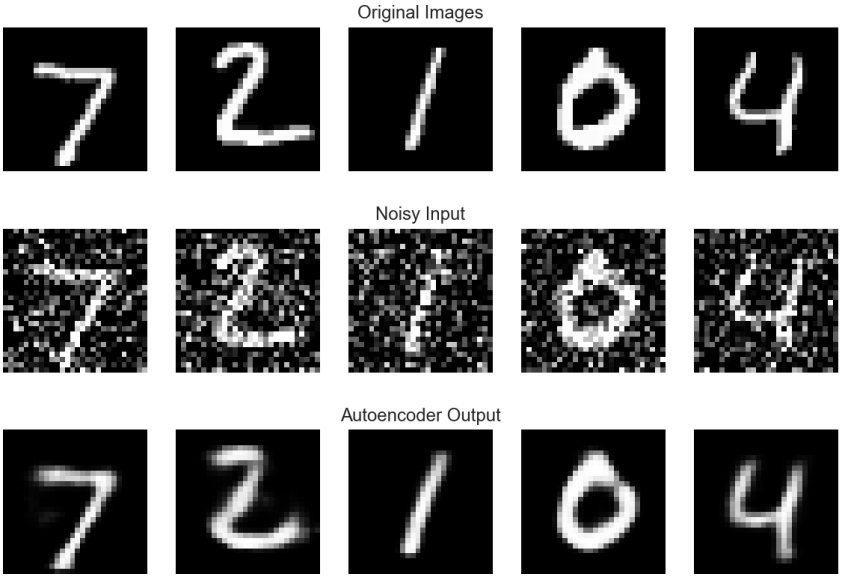
Obrázek 14: Popis autoenkodéru [39].

Tato metoda má 4 hyperparametry. Ty jsou velikost kódu, počet vrstev, počet neuronů na vrstvu a ztrátovou funkci. Je učena stejnou metodou, jako obyčejné neuronové sítě, a to je zpětnou propagací chyb [39].

Jednou důležitou vlastností je to, že tato metoda je ztrátová. Tím pádem, i když se autoenkodér snaží zrekonstruovat vstup, tak výstup bude vždycky do nějaké míry odlišný. Tohle umožňuje zbavení se šumu vstupu. Implementace zaměřené na zbavení se šumu, se moc neliší od normální implementace. Normální autoenkodér má vstup a objekt se kterým se tento vstup po průchodem sítí poměřuje totožný. U metody zbavení se šumu je na vstupu objekt se šumem a porovnávací objekt je bez šumu. Na obrázku 15 lze vidět nahoře původní vstupní data, uprostřed vstupní data se šumem a dole výstupní data [39].

Detekce anomálií se u této metody provádí přes střední kvadratickou chybu (někdy také metoda nejmenších čtverců) známá jako MSE z anglického mean squared error. Je to veličina vyjadřující přesnost odhadů pomocí střední hodnoty druhých mocnin rozdílů mezi odhadem/měřením a skutečností. Po vypočítání těchto hodnot se určí hranice, která bude rozdělovat to, co se považuje za normální a anomální. Typická rovnice MSE lze vidět na vzorci 2, kde A je vektor sledovaného objektu a B je vektor předpovězeného objektu [40].

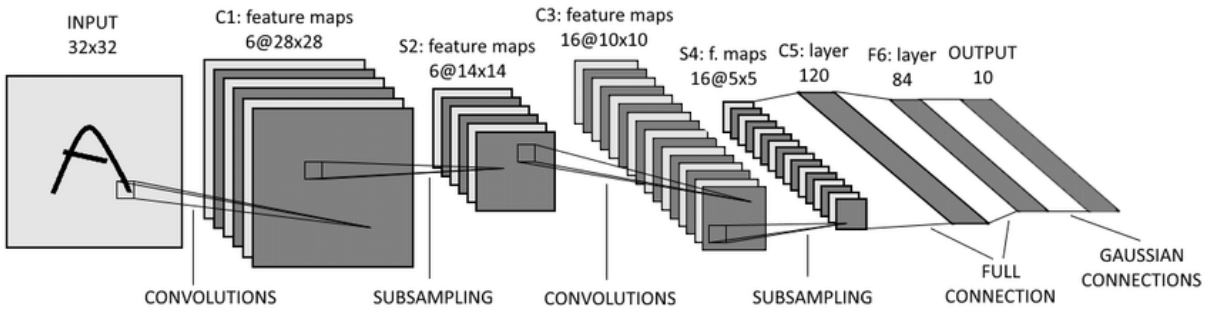
$$\sum_{i=1}^{n}(A_i - B_i)^2 \tag{2}$$



Obrázek 15: (Seshora)vstupní data, vstupní data se šumem a výstupní data u autoenkodéru [41].

4.2.5 Konvoluční neuronová síť - LeNet

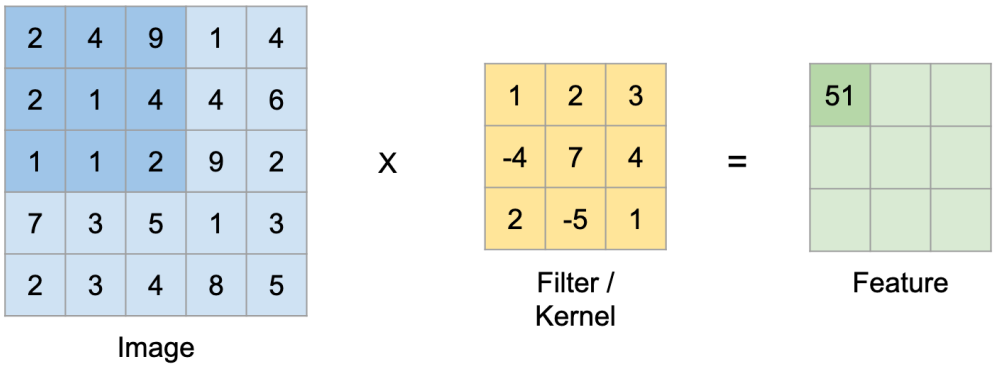
Samotná struktura neuronové sítě, která byla použita v této práci, byla implementována podle konvoluční neuronové sítě LeNet. LeNet byla vytvořena v roce 1998, jejímž autorem je Yann LeCun a kolektiv. Použitím LeNet se obvykle myslí lenet-5. Tato neuronová síť se skládá ze 2 sad konvolučních a average pooling (sdružování průměrů) vrstev, vrstvy linearizace dat do jednorozměrného vektoru, plně propojené neuronové podsítě, která se skládá ze 2 vrstev a poslední vrstva LeNet obsahující softmax klasifikátor. Tyto vrstvy jsou popsány v následující části [42].



Obrázek 16: Struktura LeNet [10].

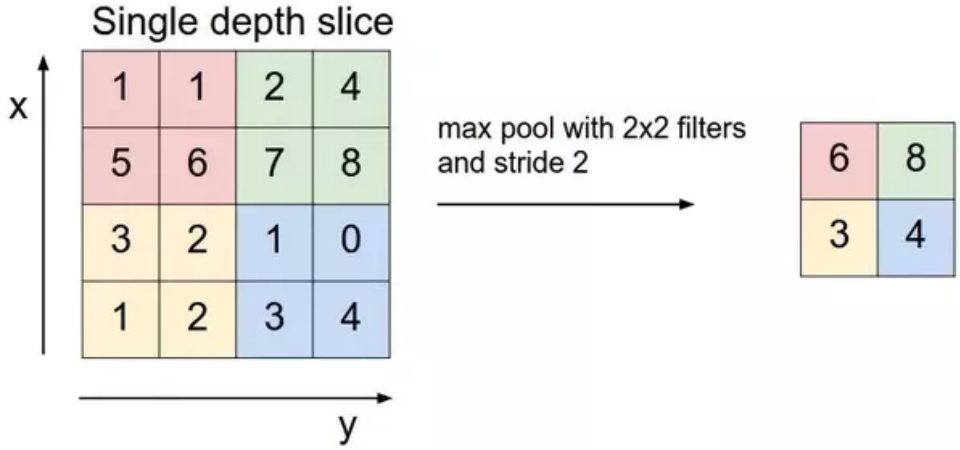
LeNet začíná konvoluční vrstvou. Konvoluce je matematická operace, kdy se ze dvou funkcí vytvoří třetí. V tomhle případě se ze vstupu a filtru vytvoří příznak. Tyto konvoluční vrstvy zmenšují matici dat, které do nich vstupují. Lze si představit, že v původní matici putuje v ní vložená menší matice, která se posouvá o jeden sloupec dokud nenarazí na poslední, a pak se

posune o řádek níže, a tohle se opakuje až do konce matice. Tahle menší matice vytváří novou výstupní matici, jejíž rozměry jsou dány, kolikrát se menší matice mohla posunout. Hodnoty jednotlivých prvků ve výstupní matici vypovídají použitím daných filtrů menší maticí. Jinými slovy tyto filtry (označované jako konvoluční jádro) aplikováním na vstup vytváří novou menší matici. Konvoluční vrstvy můžou z jedné matice udělat několik maticí, protože lze použit více filtrů pro jednu vrstvu. Když se použije konvoluční vrstva s 6 filtry, tak se ze vstupu vytvoří 6 matic. Na obrázku 17 je zobrazeno, jak to může vypadat. Tmavě modrý čtverec by se posouval a pomocí filtru by naplňoval zelenou matici napravo [43].



Obrázek 17: Obrázek funkce konvoluční vrstvy [44].

Za konvoluční vrstvou je pooling vrstva, která se používá k redukci velikosti vstupů. Tahle menší velikosti zmenší výkonovou náročnost celé sítě. Pooling vrstva rozdělí matici, která do ní vstoupila do několika částí (nastaveno parametrem), a pak provede operaci nad těmito částmi (například průměr). Výsledek této operace nad danou částí je jedno číslo, které se stane součástí nové matice. Nejčastější typy pooling vrstev jsou Max Pooling (operace nalezení maximální hodnoty), Average Pooling (operace průměru) a Sum Pooling (operace součtu). Na obrázku 18 lze vidět Max pooling vrstvu se vstupní maticí o rozměrech 4x4 a filtr rozměrů 2x2, který hledá největší hodnotu ve své části vstupní matice.



Obrázek 18: Max pooling vrstva [45].

Po 2 sadách konvoluční a pooling vrstvy následuje vrstva, která linearizuje data do jednorozměrného vektoru. Jinými slovy zploští více rozměrný vstup do jedno rozměrného vektoru. Hned za ní jsou plně propojené vrstvy, které jsou obyčejné jedno-dimenzionální vrstvy, kde každý neuron je propojen s každým neuronem následující vrstvy [43].

Poslední vrstva je zodpovědná za klasifikace a je to obyčejná vrstva s funkcí softmax. Výstupem této vrstvy je pole hodnot, které odpovídá pravděpodobnostnímu rozdělení do zvolených počtů kategorií. Součet všech hodnot v poli je roven 1, z toho vyplývá, že když se hodnota u daného prvku v poli vynásobí 100, tak se zjistí na kolik procent si neuronová síť myslí, že naučená aktivita patří danému indexu. Index v poli od softmax funkce odpovídá indexu označení aktivit. Například když ve výstupu softmax funkce má číslo na indexu 3 hodnotu 0.8, tak si algoritmus myslí na 80%, že je to aktivita kategorie s indexem 3. Tahle vrstva je zodpovědná za klasifikaci dat [46].

5 Vlastní algoritmus

Vlastní algoritmus detekce anomálií a rozpoznání lidských činností, je implementován v programovacím jazyce Python s využitím hlavně knihovny Numpy. Odkaz na stažení potřebné distribuce Anaconda a seznam potřebných balíčků je v přílohách B. Ke klasifikaci aktivit byla použita konvoluční neuronová síť a k detekci anomálií musel být použit vlastní postup, protože jiný na použitých datech nebyl úspěšný.

Jako základ byla použita konvoluční neuronová síť LeNet (kapitola 4.2.5), která sloužila ke klasifikaci jednotlivých aktivit a její výstup byl použit u detekce anomálií. V síti této práce byla ještě navíc implementována vrstva dropout, která se u LeNet neobjevuje. Dropout vrstva náhodně vybere neurony, které se při učení ignorují, ale při testování se neignorují. Množství těchto neuronů se nastavuje parametrem vrstvy. Tohle prodlužuje dobu učení k dosažení stejných výsledků, ale zvětšuje míru generalizace, která pomáhá algoritmu, aby byl více schopen rozpoznat data podobné naučeným. Bez generalizace by algoritmus měl problém rozpoznávat jiná než naučená data [47].

V této práci byly pokusy využít SVM (SVM je popsán v kapitole 4.2.1, samotné umístění programu je popsáno na B.1) a autoenkodér (autoenkodér je popsán v kapitole 4.2.4, samotné umístění programu je popsáno na B.2), ale ani jeden postup neměl vyhovující výsledky. SVM vždy ukazovalo jen jednu kategorii a autoenkodér měl hodnoty MSE, které se moc nelišily u naučených a anomálních činnosti. V následující části je popsán způsob, jakým se upravily příznaky pro neuronovou síť a způsob detekce anomálií, který vychází z výstupu neuronové sítě.

5.1 Příznaky

V této práci byly použity příznaky (viz kapitola 4.1) kloubů těla, které se získávají z hloubkových map. Tyto kostry obsahují většinou mezi 15 a 25 klouby. Mezi tyto klouby se počítají i body, které nejsou klouby samy o sobě, ale jsou důležitými body kostry. Jsou to body hlavy, středu těla (středu páteře) a krku. Každému kloubu se vytvořil normalizovaný příznak pro koordináty os x, y, z, tak aby pohyb byl invariantní vůči absolutní pozici a natočení těla. Jednotlivé příznaky se vypočítaly podle vzorce 3, který byl představen v článku [48].

$$D_i = \frac{J_i - J_0}{\|J_2 - J_0\|}, 1, 2, ..., P - 1 \tag{3}$$

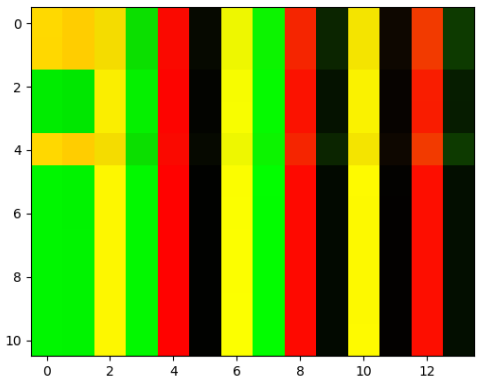
Kde P je počet kloubů, J_0 je pozice kloubu pánve a J_2 je pozice kloubu krku. D_i je příznak i-tého kloubu, který znázorňuje vektorovou vzdálenost mezi klouby J_i a J_0 (pánev), normalizované vůči vzdálenosti mezi klouby J_2 (krk) a J_0 (pánev).

Jako výsledek se získal vektor příznaků pro jednotlivé snímky. Tento vektor se dále použil jako vstup u dvou funkcí, které změnily formát těchto příznaků a tento formát se použil jako vstupní data pro konvoluční síť. U každého datasetu se použila jenom jedna funkce, která upravovala příznaky. Empiricky bylo zjištěno, která funkce funguje lépe pro daný dataset a ta se použila.

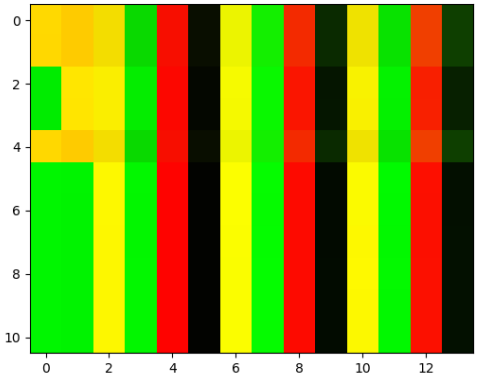
První funkce využívá funkci `poly1d` z knihovny Numpy [49], která vypočítává polynom pro příznaky každého kloubu zvlášť. Výsledek tohoto polynomu při x rovno 1 je výstup této funkce. Vstupem `poly1d` jsou příznaky jednoho kloubu seřazeny chronologicky. Druhá funkce vytvoří průměry hodnot příznaků pro každý kloub zvlášť. Datasety JHMDB, Florence 3D Actions a UTKinet-Action3D využívají funkci založenou na `poly1d` a dataset MSR Action3D využívá funkci průměrů.

Výstupem těchto funkcí jsou 3 matice, kde jedna matice představuje jednu dimenzi kloubů. V řadě matice je tolik prvků co má dataset kloubů bez 1 kloubu, protože vzniklé příznaky ze vzorce 3 se vztahují na vzdálenost od pánve, takže pro kloub pánve se nevytvoří příznak. Hodnota prvku je vypočítaná pomocí funkce `poly1d` nebo pomocí vypočítání průměru. Tyto funkce byly zmíněny v předchozím odstavci. Počet řad v matici je dán parametrem. Tento parametr určuje množství použitých vrstev, kde jeho hodnota je rovna počtu použitých vrstev. Každá vrstva vytvoří $2x + 1$ řad, kde x je počet řad předchozí vrstvy a počáteční vrstva má pouze 1 řadu. Každá řada bere v potaz pouze $\frac{1}{(2y)}$ délky aktivity, kde y je počet pořadové číslo předchozí vrstvy. Každá další řada má posunutý začátek od kdy začíná brát data, jak lze vidět na obrázku 20. Řady všech vrstev dohromady dají výstup funkce. Ukázka těchto matic reprezentovaných pomocí RGB kanálů, jejich podobnost mezi stejnými kategoriemi a odlišnost mezi odlišnými kategoriemi je vyobrazena na obrázku 19. Kde osa x značí počet kloubových příznaků a osa y značí kolikrát se provedla funkce na daty (respektive je to počet řad, které jsou vysvětlené v předchozí části).

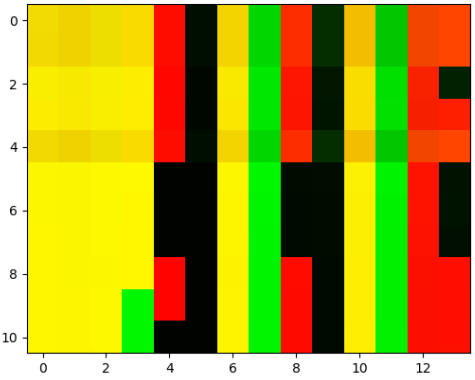
Parametr byl vyzkoušen až po hodnotu 5, kdy už nedocházelo ke zlepšení výsledků u žádného z použitých datasetů. Na obrázku 20 je vyobrazeno, která řada si bude brát kterou část dat, při parametru hodnoty 3.



(a) 1. vzorek aktivity česání vlasů.

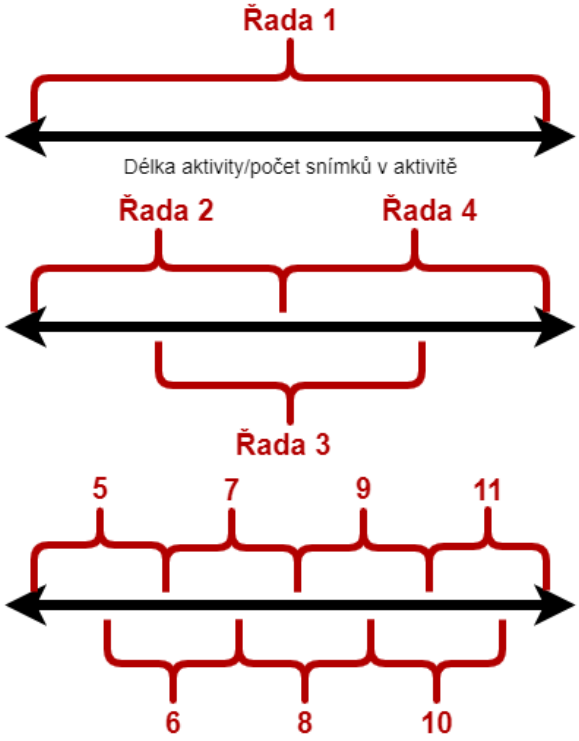


(b) 2. vzorek aktivity česání vlasů



(c) Vzorek aktivity shybu.

Obrázek 19: Grafické zobrazení matic vstupujících do neuronové sítě. Vzorky jsou z datasetu JHMDB.



Obrázek 20: Počet řádků a označení jakou část z celkové délky snímků řádek obsahuje.

5.2 Detekce anomálií

Detekce anomálií je u různých forem dat dobře probádaná. Například pro detekce anomálií u méně komplexní dat, nebo u statických obrázků existují metody, které byly mnohokrát otestovány a byla prokázána jejich funkčnost.

U lidských činností je dnes klasifikace dostatečně probádaná a existuje mnoho způsobů, jak ji implementovat. U detekce anomálií těchto činností je tomu naopak. Není mnoho článků, které se tímto zabývají,a proto je výběr metody pro detekci anomálií složitý.

V této práci byl pokus o použití autoenkodéru pro detekci anomálií, ale výsledky nebyly uspokojující. Hodnoty MSE (střední kvadratická chyba) u natrénovaných a testovaných dat byly velmi podobné a přesnost u trénovacích dat byla nízká. Tedy tato metoda v téhle práci nebyla schopna detekce anomálií nad datasety zmíněných v kapitolách 3.1 až 3.4.

Z výše uvedených důvodů byla vytvořena metoda, která je založena na observaci výsledků poskytnutých klasifikátorem konvoluční neuronové sítě. Aktivity které algoritmus úspěšně poznal, tak měly většinou vysokou pravděpodobnost pouze u jedné kategorie. Anomálie měly vetšinou 2 nebo více kategorií, které měly velkou pravděpodobnost, a to většinou přes 10%. Počet těchto kategorií, které mají mít pravděpodobnost větší, než je daná mez byl pro každý dataset empiricky zjištěn, a to samé platí pro hodnotu této meze.

6 Experimenty

Algoritmus byl vyvinut ve vývojovém prostředí PyCharm za pomocí distribuce Anaconda jazyka Python (kapitola 2). K experimentům byl využit počítač s operačním systém Windows 10 64-bit a tímto hardwarem:

- CPU: Intel Core i5-4690K 3,50GHz
- GPU: Nvidia GeForce GTX 970 4GB, firma MSI, verze ovladačů 442,19
- RAM: 8GB

U datasetů byly pro klasifikaci provedeny 2 typy testů, kdy u prvního se použila 1/3 dat pro učení a 2/3 pro testování a u druhého se použily 2/3 dat pro učení a 1/3 pro testování. U datasetů MSR Action3D a JHMDB se toto rozdělení provedlo pomocí epizod. Epizody v tomhle případě znamenají, že každý subjekt provedl činnost vícekrát, kde každé provedení značí jednu epizodu. Oba tyhle datasety mají 3 epizody pro každou aktivitu, tak se to rozdělilo tak, že u prvního typu se použila 1 epizoda na učení a 2 na testování a u druhého typu naopak. Dataset UTKinect-Action3D a Florence 3D Actions nejsou rozděleny do 3 epizod, tak se rozdělení na třetiny provedlo rozdělením podle počtu subjektů. Oba datasety mají 10 subjektů, kteří vykonávají aktivity, takže u prvního experimentu se použily subjekty 1,2,3 na učení a 4,5,6,7,8,9,10 na testování a u druhého experimentu 1,2,3,4,5,6,7 na učení a 8,9,10 na testování. Oba typy testů se provedly 3 krát kvůli zjištění, jestli každé spuštění negeneruje odlišné výsledky.

Klasifikace byla provedena pomocí konvoluční neuronové sítě Tensorflow, která byla zakončená klasifikační softmax funkcí. Výsledky experimentů jsou interpretovány měřítky, které jsou vypočítány následujícími rovnicemi [50].

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$PPV = \frac{TP}{TP + FP} \tag{6}$$

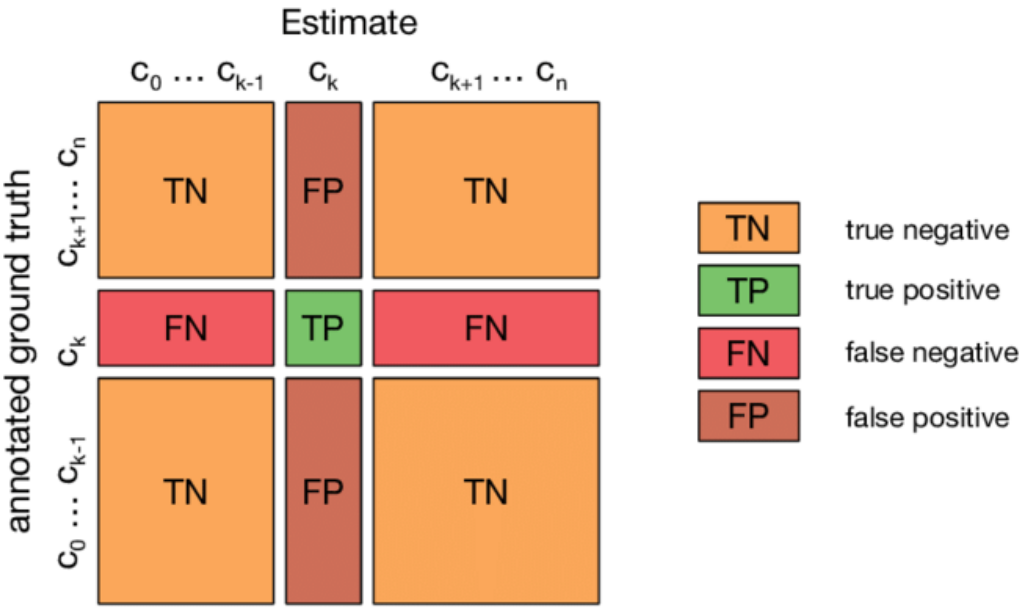
$$F1 = \frac{2 * TPR * PPV}{TPR + PPV} \tag{7}$$

Kde ACC značí přesnost, TPR úplnost, PPV preciznost a F1 F skóre. U těchto rovnic jsou použité proměnné TP (True positive - správný pozitivní), TN (True negative - správný negativní), FP (false positive - nesprávný pozitivní), FN (False negative - nesprávný negativní), které jsou vysvětlené v následující části: [50]

- TP: Objekt je pozitivní a byl rozpoznán jako pozitivní
- FN: Objekt je pozitivní a byl rozpoznán jako negativní
- TN: Objekt je negativní a byl rozpoznán jako negativní
- FP: Objekt je negativní a byl rozpoznán jako pozitivní

Všechny tyto rovnice vyjadřují úspěšnost klasifikace (predikce) v koeficientu procent, ale přesnost, úplnost a preciznost mají problémy. Pokud je vysoká úplnost a nízká preciznost, tak byla většina správných případu korektně rozpoznána, ale patří mezi ně i velké množství případů, kdy byly případy rozpoznány jako správné, ale ve skutečnosti byly nesprávné. Když je situace opačná, že je nízká úplnost a vysoká preciznost, tak bylo málo správně rozpoznaných případů, ale patří mezi ně málo případů, které byly rozpoznány jako správné, ale byly ve skutečnosti špatné. Přesnost ve většině případů bývá vysoká a nic moc neříkající. Proto se používá hlavně F skóre, které bere potaz poměr úplnosti a preciznosti.

Na obrázku 21 lze vidět matici záměn pro problematiku s více kategoriemi. Zelená označuje správně označené objekty, červená kdy vybraný objekt C_k nebyl rozpoznán, když měl být rozpoznán, hnědá kdy objekt byl nesprávně rozpoznán a žlutá kdy objekt neměl být rozpoznán a nebyl rozpoznán.



Obrázek 21: Matice záměn pro problematiku s více kategoriemi [51].

6.1 Výsledky

Následující tabulky zaznamenávají data klasifikací při 1/3 dat použitých na učení a 2/3 dat použitých na klasifikaci. Pro daná měření jsou zde zobrazeny hodnoty všech rovnic, které byly popsány v předchozí části na vzorcích 4 až 7 a čas, jak dlouho algoritmus běžel.

Kinect	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,88	0,24	0,23	0,23	9,7
2. měření	0,87	0,26	0,25	0,26	9,5
3. měření	0,87	0,26	0,21	0,23	9,6

Tabulka 1: Klasifikace pro UTKinect Action3D dataset při 1/3 data na učení a 2/3 na testování

Florence	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,89	0,40	0,44	0,39	14,4
2. měření	0,89	0,39	0,49	0,40	14,1
3. měření	0,89	0,40	0,46	0,41	14,2

Tabulka 2: Klasifikace pro Florence 3D Actions dataset při 1/3 data na učení a 2/3 na testování

JHMDB	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,99	0,84	0,84	0,84	94,6
2. měření	0,99	0,83	0,83	0,83	69,4
3. měření	0,99	0,84	0,84	0,83	70,9

Tabulka 3: Klasifikace pro JHMDB dataset při 1/3 data na učení a 2/3 na testování

MSR	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,97	0,67	0,70	0,67	17,6
2. měření	0,97	0,66	0,69	0,67	10,9
3. měření	0,97	0,69	0,71	0,69	10,9

Tabulka 4: Klasifikace pro MSR Action3D dataset při 1/3 data na učení a 2/3 na testování

Následující tabulky zaznamenávají data klasifikací při 2/3 dat použitých na učení a 1/3 dat použitých na klasifikaci. Pro daná měření jsou zde zobrazeny hodnoty všech rovnic, které byly popsány v předchozí části 4 5 6 7 a čas, jak dlouho algoritmus běžel.

Kinect	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,87	0,29	0,35	0,27	10,6
2. měření	0,88	0,30	0,34	0,28	10,7
3. měření	0,87	0,27	0,38	0,28	10,5

Tabulka 5: Klasifikace pro UTKinect Action3D dataset při 2/3 data na učení a 1/3 na testování

Florence	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,90	0,47	0,46	0,42	14,1
2. měření	0,89	0,43	0,47	0,42	14,3
3. měření	0,90	0,50	0,48	0,45	14,2

Tabulka 6: Klasifikace pro Florence 3D Actions dataset při 2/3 data na učení a 1/3 na testování

JHMDB	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,99	0,89	0,89	0,88	70,5
2. měření	0,99	0,89	0,89	0,89	68,4
3. měření	0,99	0,90	0,90	0,90	68,9

Tabulka 7: Klasifikace pro JHMDB dataset při 2/3 data na učení a 1/3 na testování

MSR	Přesnost[%]	Úplnost[%]	Preciznost[%]	F skóre[%]	čas[s]
1. měření	0,98	0,74	0,75	0,75	12,2
2. měření	0,98	0,77	0,78	0,77	11,7
3. měření	0,98	0,77	0,79	0,77	11,7

Tabulka 8: Klasifikace pro MSR Action3D dataset při 2/3 data na učení a 1/3 na testování

Z tabulek lze vidět, že množství dat použitých na naučení má vliv na výsledky klasifikace a výsledky experimentů jsou stabilní při vícenásobné zapnutí, nad stejnými daty. Druhý experiment, kde byly použity 2/3 dat na učení měl lepší výsledky F skóre a to pro Kinect 28%, pro Florence 43%, pro JHMDB 89% a pro MSR 76%. Tyhle výsledky jsou pro JHMDB a MSR dobré, avšak pro Kinect a Florence špatné.

Pro detekci anomálií byly využity modely, kdy byly použity 2/3 dat na učení a 1/3 na testování. Postupně se při experimentech přidával počet anomálních kategorií a měřil se počet správně detekovaných anomálií, poměr správně zachycených anomálií k celkovému počtu anomálií, počet nesprávně detekovaných anomálií a F skóre. Výsledky byly zaznamenány do grafů, které jsou k nalezení v přílohách A. Z výsledků lze vidět, že zvolená metoda detekce závisí na množství kategorií, která jsou vybrána jako anomálie. Pro všechny datasety se krom JHMDB

poměr správně detekovaných anomálií vůči celkovému počtu anomálií pomalu snižoval, ale taky se rychle snižoval počet aktivit nesprávně detekovaných jako anomálie.

Dataset JHMDB byl na rozdíl od ostatních celkem stabilní. Zde se F skóre drží dlouho hodnot kolem $80\% - 90\%$ až ke konci osy kde prudce klesá. Poměr správně detekovaných anomálií vůči celkovému počtu anomálií se držel kolem hodnoty 50%, až do doby, kdy program neměl dost kategorií na naučení a výsledky začaly být ke konci extrémně klesat, jak lze vidět v přílohách A. Tenhle fakt, že při malém počtu naučených kategorií se skokově propadají výsledky detekce anomálií a F skóre platí pro všechny datasety.

7 Závěr

Cílem této práce bylo se seznámit s problematikou klasifikace a detekce anomálií u lidských činností, za použití příznaků kloubů z lidské kostry. Dále získané znalosti implementovat a otestovat.

Na učení a testování byly použity 4 datasety: UTKinect-Action3D, Florence 3D Actions, JHMDB a MSR Action3D. Na nich byla provedena klasifikace kategorií za pomocí konvoluční neuronové sítě, která využívala architekturu sítě LeNet.

Výsledky klasifikace aktivit se u datasetů JHMDB a MSR Action3D ukázaly, jako velmi dobré s přesností $75 - 90\%$. Dále byl dataset Florence 3D Actions s přesností kolem 50%, kde tenhle výsledek není moc dobrý, ale je stále lepší, než výsledek datasetu UTKinect-Action3D, kde se dosáhlo jen kolem 30% úspěšnosti.

Detekce anomálií u lidských aktivit se ukázala, jako složitá problematika, která není moc prozkoumána na rozdíl od klasifikace těchto aktivit, která už má několik metod, jak ní přistupovat.

Použitá metoda pro detekci anomálií v této práci ukázala, že úspěšnost záleží na množství anomálních kategorií, a že JHMDB obsahuje nejkvalitnější data, protože jak u klasifikace tak i u detekce anomálií vykazoval vysokou a stabilní úspěšnost. Detekce anomálií u zbylých datasetů měla s vyšším počtem anomálií nižší úspěšnost zachycení anomálií, ale taky skokově klesaly případy, kdy byly nesprávně zachyceny aktivity jako anomálie.

Literatura

1. *What is Python? Executive Summary | Python.org. Welcome to Python.org* [online] [cit. 2020-04-19]. Dostupné z: https://www.python.org/doc/essays/blurb/.
2. *Why Anaconda for Data Science?. Anaconda | The World's Most Popular Data Science Platform* [online] [cit. 2020-04-11]. Dostupné z: https://www.anaconda.com/anaconda-community/.
3. *Svobodný a otevřený software* [online] [cit. 2020-04-11]. Dostupné z: https://cs.wikipedia.org/wiki/Svobodn%C3%BD_a_otev%C5%99en%C3%BD_software.
4. *Package, dependency and environment management for any language Platform* [online] [cit. 2020-04-11]. Dostupné z: https://conda.io/en/latest/.
5. *The Python Package Index* [online] [cit. 2020-04-11]. Dostupné z: https://pypi.org/project/pip/.
6. *Anaconda (Python distribution) - Wikipedia.* [online] [cit. 2020-04-11]. Dostupné z: https://en.wikipedia.org/wiki/Anaconda_(Python_distribution).
7. *Tensorflow - Wikipedia* [online] [cit. 2020-04-11]. Dostupné z: https://en.wikipedia.org/wiki/TensorFlow.
8. *Google's Tensor Processing Unit explained: this is what the future of computing looks like* [online] [cit. 2020-04-11]. Dostupné z: https://www.techradar.com/news/computing-components/processors/google-s-tensor-processing-unit-explained-this-is-what-the-future-of-computing-looks-like-1326915.
9. *Large-Scale Machine Learning on Heterogeneous Distributed Systems* [online] [cit. 2020-04-11]. Dostupné z: http://download.tensorflow.org/paper/whitepaper2015.pdf.
10. *Google Developers* [online] [cit. 2020-04-11]. Dostupné z: https://developers.google.com/machine-learning/crash-course/images/TFHierarchyNew.svg.
11. *Why use Keras - Keras Documentation. Home - Keras Documentation* [online] [cit. 2020-04-11]. Dostupné z: https://keras.io/why-use-keras/.
12. *Kinect - Wikipedia* [online] [cit. 2020-04-19]. Dostupné z: https://en.wikipedia.org/wiki/Kinect#Kinect_for_Windows_(2012).
13. *Jameco Electronics - Electronic Components Distributor* [online] [cit. 2020-04-19]. Dostupné z: https://www.jameco.com/Jameco/workshop/howitworks/xboxkinect-fig4.jpg.
14. *How Does The Kinect 2 Compare To The Kinect 1?. Augmented Reality and Natural User Interface Experiences* [online] [cit. 2020-04-19]. Dostupné z: http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1.
15. *Dr. J.K. Aggarwal - UTKinect-Action3D Dataset* [online] [cit. 2020-04-19]. Dostupné z: http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html.

16. XIA, L.; CHEN, C.C.; AGGARWAL, JK. View invariant human action recognition using histograms of 3D joints. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. 2012, s. 20–27.

17. *MICC - Media Integration and Communication Center. MICC - Media Integration and Communication Center* [online] [cit. 2020-04-19]. Dostupné z: `https://www.micc.unifi.it/resources/datasets/florence-3d-actions-dataset/`.

18. L. Seidenari adn V., Varano; S., Berretti; A., Del Bimbo; PALA., P. Recognizing Actions from Depth Cameras as Weakly Aligned Multi-Part Bag-of-Poses. 2013.

19. JHUANG, H.; GALL, J.; ZUFFI, S.; SCHMID, C.; BLACK, M. J. Towards understanding action recognition. In: *International Conf. on Computer Vision (ICCV)*. 2013-12, s. 3192–3199.

20. *JHMDB Dataset* [online] [cit. 2020-04-19]. Dostupné z: `http://jhmdb.is.tue.mpg.de/`.

21. *Zicheng Liu - MSR Dataset* [online] [cit. 2020-04-19]. Dostupné z: `https://documents.uow.edu.au/~wanqing/MSRActionRecognitionDatasetsCodes%20-%20Zicheng.htm`.

22. *activity picture.* [online] [cit. 2020-04-19]. Dostupné z: `https://ars.els-cdn.com/content/image/1-s2.0-S016786551830045X-gr1.jpg`.

23. *Unsupervised learning - Wikipedia* [online] [cit. 2020-04-19]. Dostupné z: `https://en.wikipedia.org/wiki/Unsupervised_learning`.

24. *Supervised learning - Wikipedia* [online] [cit. 2020-04-19]. Dostupné z: `https://en.wikipedia.org/wiki/Supervised_learning`.

25. *Semi-supervised learning - Wikipedia* [online] [cit. 2020-04-19]. Dostupné z: `https://en.wikipedia.org/wiki/Semi-supervised_learning`.

26. *Support vector machines – Wikipedie* [online] [cit. 2020-04-19]. Dostupné z: `https://cs.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_neuronov%C3%A1_s%C3%AD%C5%A5`.

27. *File:Svm max sep hyperplane with margin.png - Wikimedia Commons* [online] [cit. 2020-04-19]. Dostupné z: `https://en.wikipedia.org/wiki/File:SVM_margin.png`.

28. *Kernel Methods, How it Works, Types of Kernel with Appropriate Equation. Best Online Training and Video Courses, eduCBA* [online] [cit. 2020-04-19]. Dostupné z: `https://www.educba.com/kernel-methods/`.

29. *File:SeparationDone.png - Educba* [online] [cit. 2020-04-19]. Dostupné z: `https://cdn.educba.com/academy/wp-content/uploads/2019/08/SeparationDone.jpg`.

30. *k-nearest neighbors algorithm - Wikipedia.* [online] [cit. 2020-04-19]. Dostupné z: `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`.

31. *1.6. Nearest Neighbors — scikit-learn 0.22.2 documentation. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation* [online] [cit. 2020-04-19]. Dostupné z: `https://scikit-learn.org/stable/modules/neighbors.html`.

32. *File:KnnClassification.svg - wikipedia* [online] [cit. 2020-04-19]. Dostupné z: https://en.wikipedia.org/wiki/File:KnnClassification.svg/.

33. *2.3. Clustering — scikit-learn 0.22.2 documentation. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation* [online] [cit. 2020-04-19]. Dostupné z: https://scikit-learn.org/stable/modules/clustering.html.

34. *k-means clustering - Wikipedia.* [online] [cit. 2020-04-19]. Dostupné z: https://en.wikipedia.org/wiki/K-means_clustering.

35. *ResearchGate | Find and share research* [online] [cit. 2020-04-19]. Dostupné z: https://www.researchgate.net/profile/Patricio_Benavente/publication/318042263/figure/fig1/AS:662908132397056@1535061023270/K-means-example-in-2D-space-a-Cluster-centroids-are-initialized-at-random-positions.png.

36. *Towards data science - Artificial Neural Networks* [online] [cit. 2020-04-19]. Dostupné z: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6#04e7.

37. *What is the Difference Between a Parameter and a Hyperparameter?. Machine Learning Mastery* [online] [cit. 2020-04-27]. Dostupné z: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/g.

38. *Towards data science - Artificial Neural Networks - Picture* [online] [cit. 2020-04-19]. Dostupné z: https://miro.medium.com/max/1400/1*Gh5PS4R_A5drl5ebd_gNrg@2x.png.

39. *Towards data science - Autoencoders* [online] [cit. 2020-04-19]. Dostupné z: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

40. *Mean squared error - Wikipedia.* [online] [cit. 2020-04-27]. Dostupné z: https://en.wikipedia.org/wiki/Mean_squared_error.

41. *Towards data science - Autoencoders - Picture* [online] [cit. 2020-04-27]. Dostupné z: https://miro.medium.com/max/1400/1*hfzos8xmCGjrgpTW78PFLg@2x.png.

42. *Yann LeCun's Home Page* [online] [cit. 2020-04-11]. Dostupné z: http://yann.lecun.com/exdb/lenet/.

43. *Understanding of Convolutional Neural Network (CNN) — Deep Learning. Medium – Get smarter about what matters to you.* [online] [cit. 2020-04-11]. Dostupné z: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

44. *Towards data science - Conovoltuion - Picture* [online] [cit. 2020-04-27]. Dostupné z: https://miro.medium.com/max/3412/1*xBkRA7cVyXGHIrtngV3qlg.png.

45. *Pooling vrstva* [online] [cit. 2020-04-11]. Dostupné z: https://miro.medium.com/max/1204/1*SmiydxM5lbTjoKWYPiuzWQ.png.

46. *A Simple Explanation of the Softmax Function - victorzhou.com.* [online] [cit. 2020-04-27]. Dostupné z: https://victorzhou.com/blog/softmax/.

47. *Dropout Regularization in Deep Learning Models With Keras. Machine Learning Mastery* [online] [cit. 2020-04-28]. Dostupné z: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/.

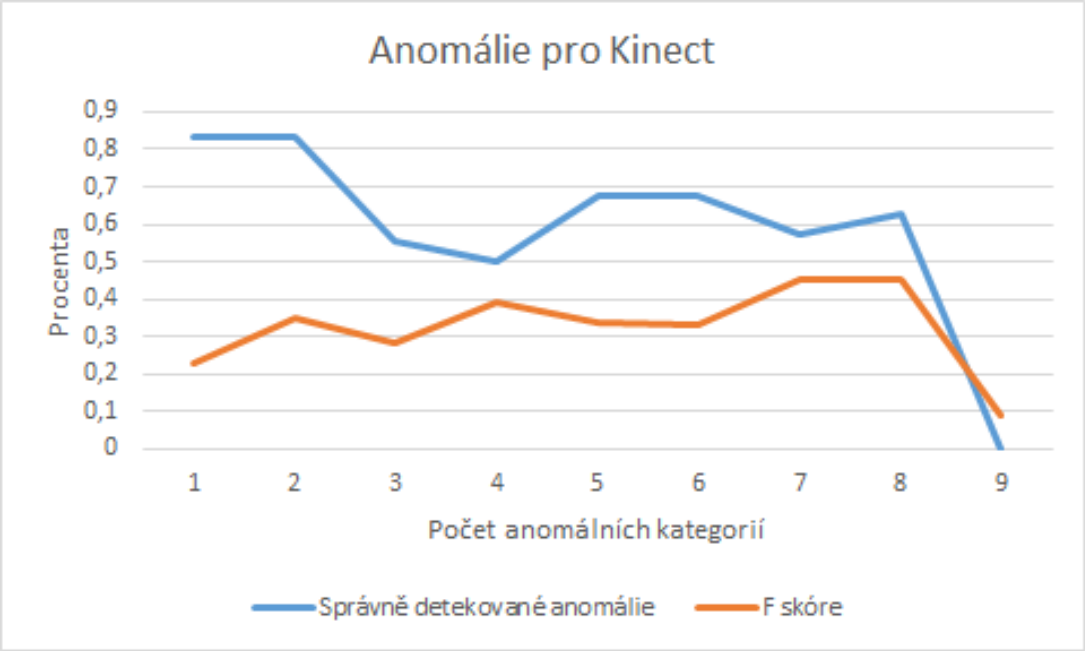
48. *A Human Activity Recognition System Using Skeleton Data from RGBD Sensors. Publishing Open Access research journals and papers | Hindawi* [online] [cit. 2020-04-27]. Dostupné z: https://www.hindawi.com/journals/cin/2016/4351435/.

49. *numpy.poly1d — NumPy v1.17 Manual* [online] [cit. 2020-04-27]. Dostupné z: https://docs.scipy.org/doc/numpy/reference/generated/numpy.poly1d.html.

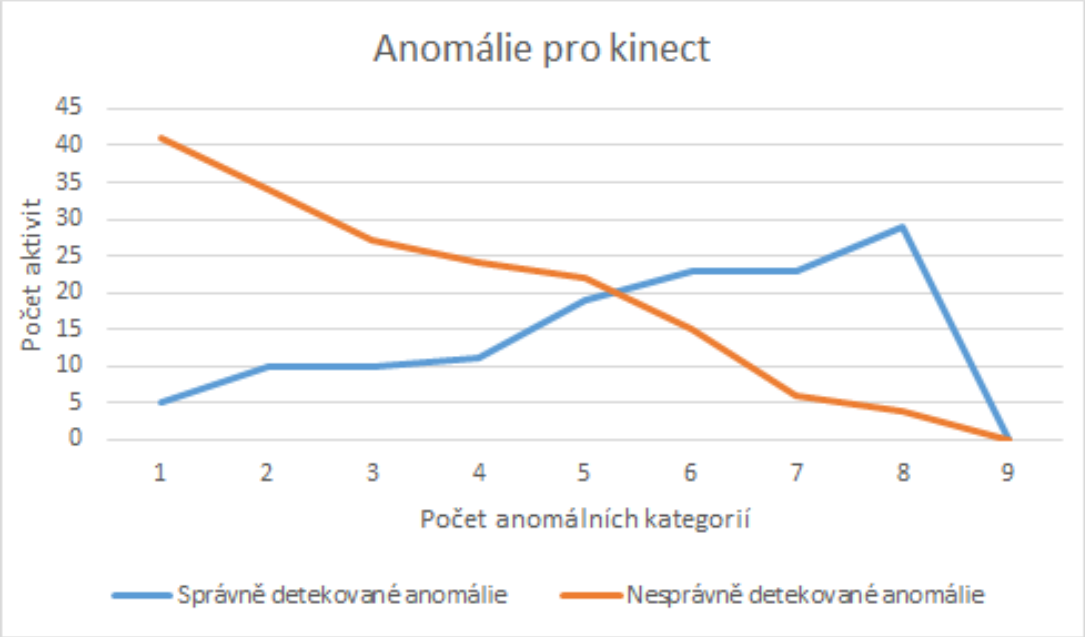
50. *Confusion Matrix in Machine Learning - GeeksforGeeks. GeeksforGeeks | A computer science portal for geeks* [online] [cit. 2020-04-28]. Dostupné z: https://www.geeksforgeeks.org/confusion-matrix-machine-learning//.

51. *ResearchGate | Find and share research - Confusion Matrix* [online] [cit. 2020-04-28]. Dostupné z: https://www.researchgate.net/profile/Frank_Krueger2/publication/314116591/figure/fig7/AS:614085901185031@1523420896093/Confusion-matrix-for-multi-class-classification-The-confusion-matrix-of-a.png.

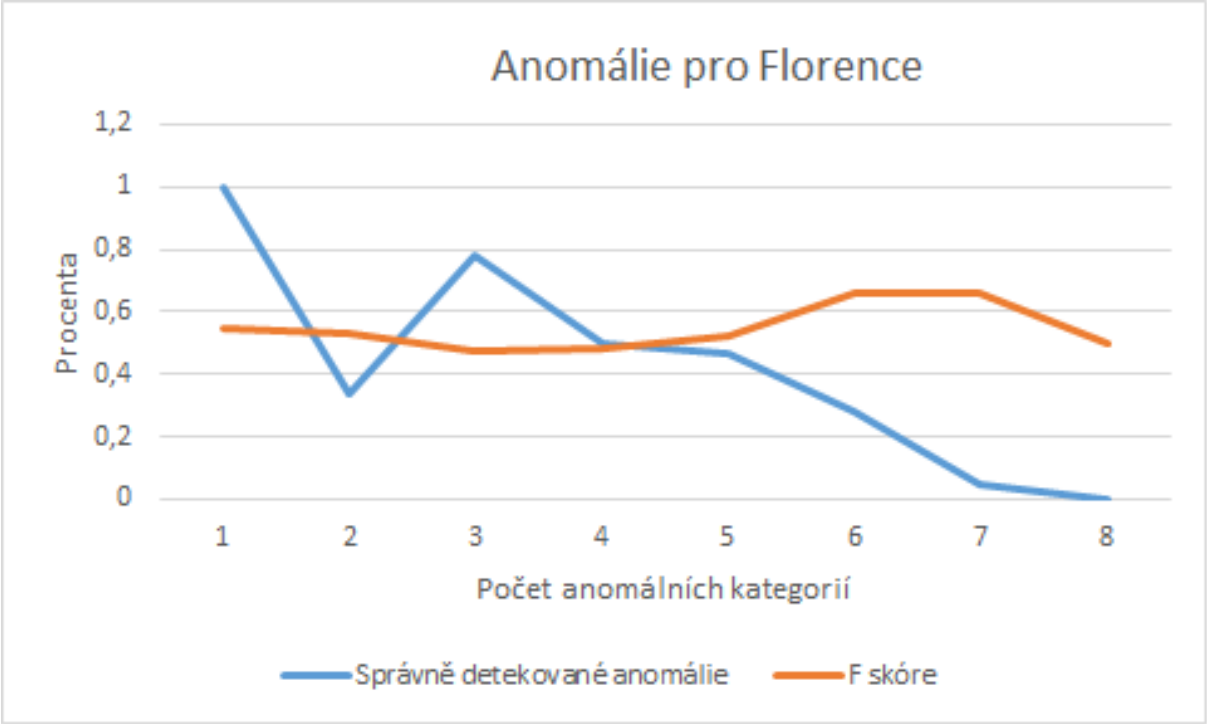
A Grafy pro detekci anomálií



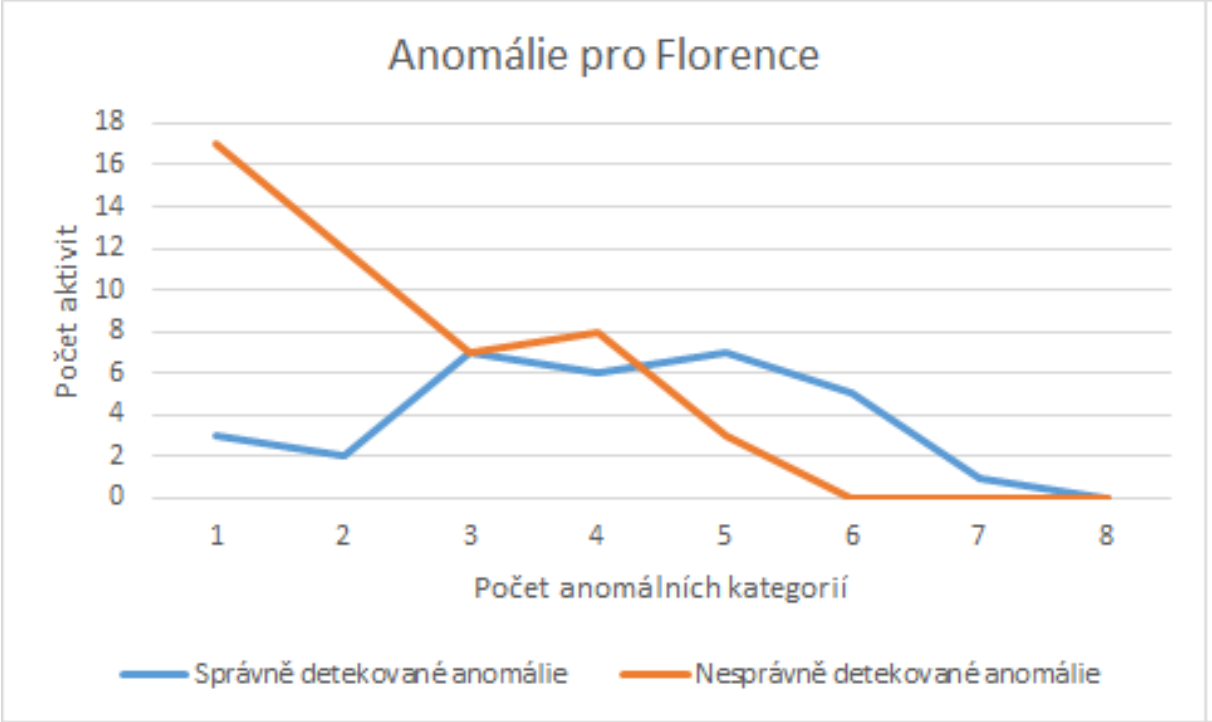
Obrázek 22: Procentuální detekce anomálií pro UTKinect Action3D dataset



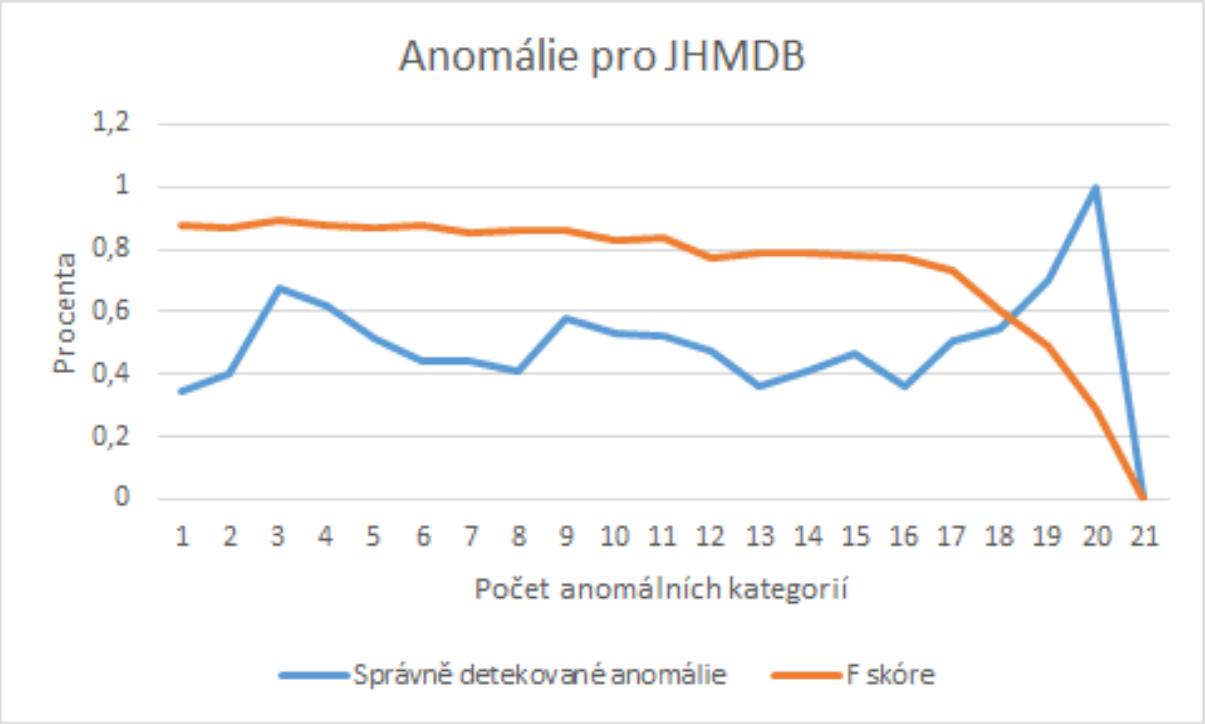
Obrázek 23: Čistá detekce anomálií pro UTKinect Action3D dataset



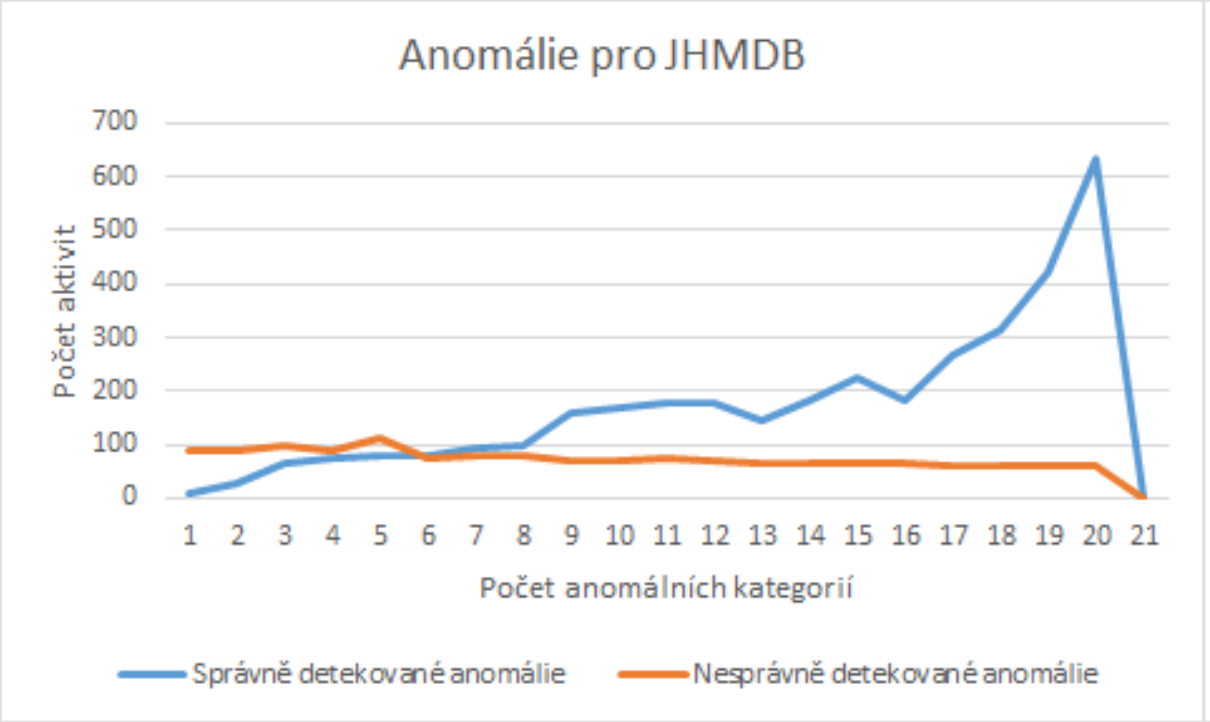
Obrázek 24: Procentuální detekce anomálií pro Florence 3D actions dataset



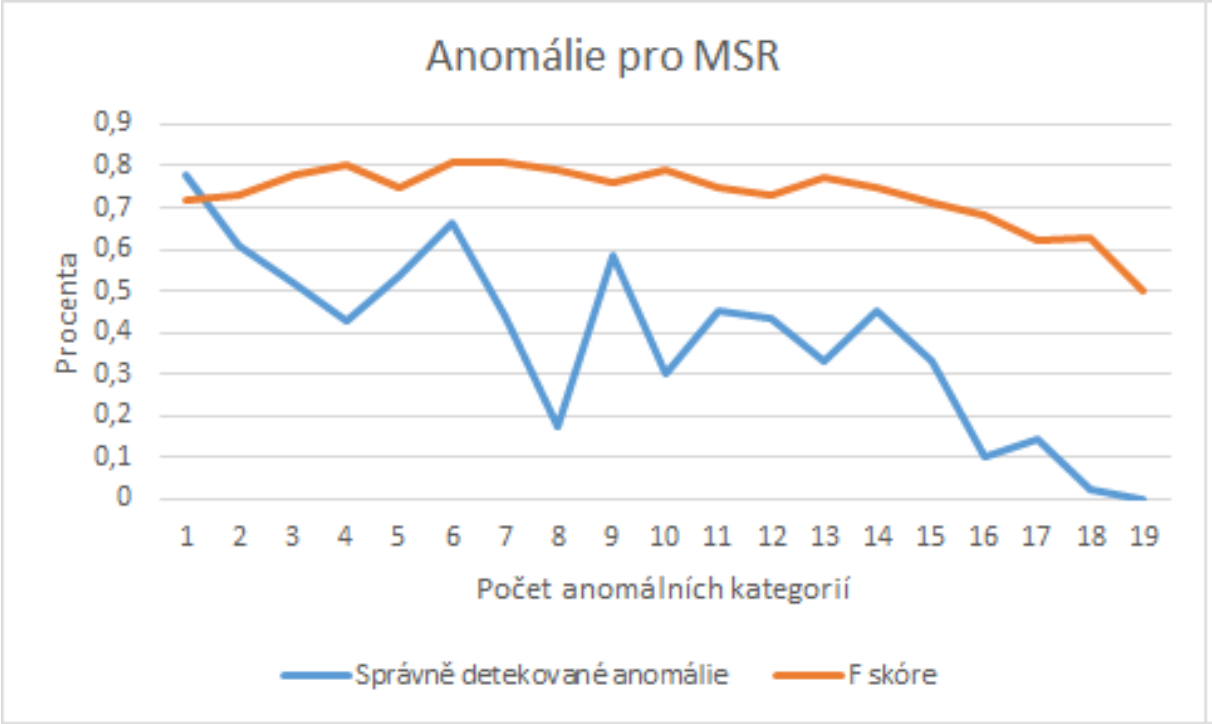
Obrázek 25: Čistá detekce anomálií pro Florence 3D actions dataset



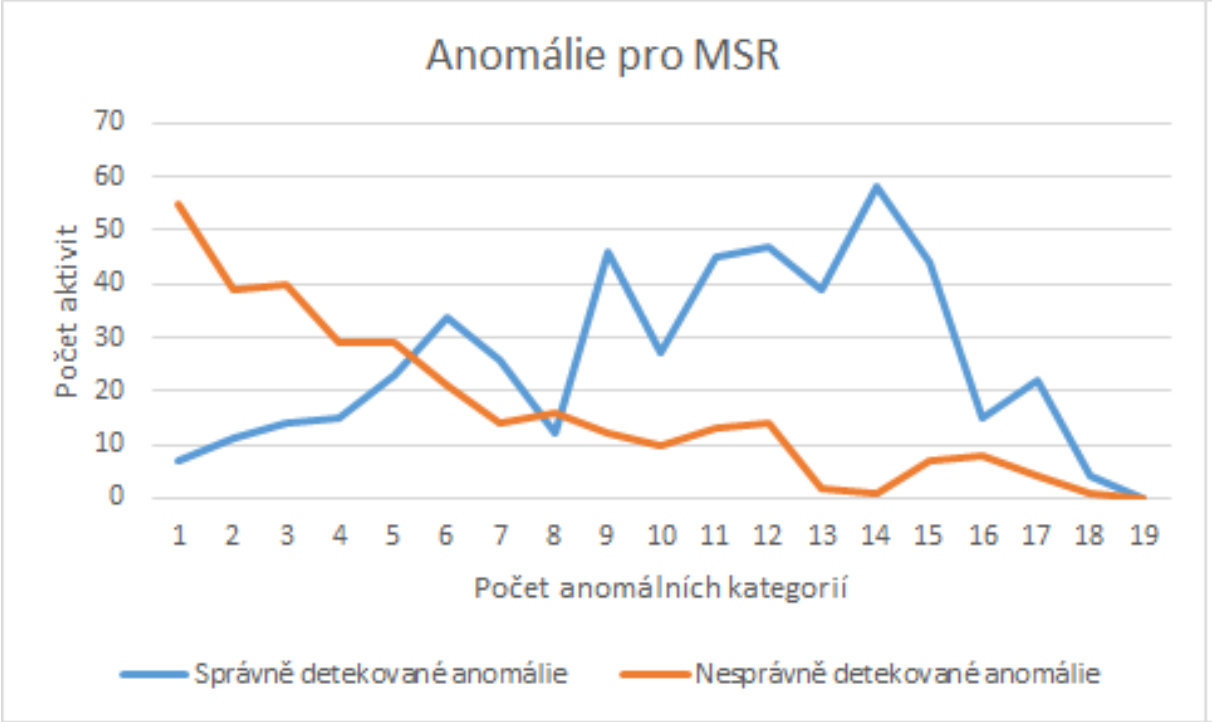
Obrázek 26: Procentuální detekce anomálií pro JHMDB dataset



Obrázek 27: Čistá detekce anomálií pro JHMDB dataset



Obrázek 28: Procentuální detekce anomálií pro MSR Action3D dataset



Obrázek 29: Čistá detekce anomálií pro MSR Action3D dataset

B Adresáře příloh

Tyto soubory se nacházejí v souboru `2020_STE03270_BP_prilohy.zip`, který je přiložen k této práci. K rozjetí programu je třeba distribuce pythonu Anaconda, která je dostupná na https://www.anaconda.com/products/individual. S ní je potřeba vytvořit prostředí, kde je zapotřebí nainstalovat balíčky tensorflow 2.0, opencv (3.4.2), matplotlib (3.1.3), numpy (1.18.1). Anaconda sama zajistí kompatibilitu verzí těchto balíčků.

B.1 SVM

`prilohy/SVM/SVM.py`

B.2 Autoenkodér

`prilohy/autoencoder.py`